VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Digitální syntetizátor

Digital Synthesizer

2014

Tomáš Janota

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Tomáš Janota**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Digitální syntetizátor
Digital Synthesizer

Zásady pro vypracování:

Cílem práce je navrhnout a implementovat digitální syntetizátor.

1. Nastudujte teorii digitální syntézy zvuku a obecně popište její principy.
2. Popište alespoň 3 způsoby generování zvuku v reálném čase dle vlastního výběru (např. aditivní syntéza, frekvenční modulace, fázová modulace).
3. Vyberte si jeden (vámi nastudovaný a popsaný) způsob a naimplementujte jej včetně sady metod (alespoň 5) pro modifikaci generované zvukové vlny (hlasitost a úpravu barvy zvuku).Konkrétní metody budou vámi popsány a voleny dle druhu vybraného typu syntézy.
4. Vytvořte grafické uživatelské rozhraní pro ovládání syntetizéru.

Seznam doporučené odborné literatury:

[1] The Audio Programming Book, Boulanger R., ISBN: 0262014467, 2010.
[2] Designing Sound, Farnell A., ISBN: 0262014416, 2010.
[3] Welsh's Synthesizer Cookbook: Synthesizer Programming, Sound Analysis, and Universal Patch Book, Wels F., ASIN: B000ERHA4S, 2006.
[4] Sound design, Teocharisová V., ISBN: 9788086253534, 2009.
[5] The Theory and Technique of Electronic Music, Puckette M.,ISBN: 9812700773, 2007.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Lukáš Vích**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostravě 7. května 2014 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2014 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, protože bez nich by tato práce nevznikla.

Abstrakt

Cílem této práce je popsat některé základní metody generování zvuku v digitálních hudebních syntezátorech. Dále také popsat, jak upravovat vlastnosti tohoto vygenerovaného zvuku, jako jsou například barva a hlasitost, pomocí některých základních nástrojů běžně se vyskytujících v reálných digitálních syntezátorech. A samozřejmě popsat, jak tyto nástroje fungují. A poté využít těchto znalostí k implementaci real-time digitálního softwarového FM syntezátoru, který kromě samotné FM syntézy bude implementovat i tyto základní způsoby úprav vygenerovaného tónu.

Klíčová slova: digitální syntéza, FM syntéza, hudební syntezátor

Abstract

Goal of this work is to describe some of the basic methods to generate sound in digital music synthesizers. Furthermore, to describe how to change characteristics of this generated sound, like for example tone color and volume, with some basic tools that are commonly found on real digital synthesizers. And of course to describe how these tools works. And then use this knowledge to implement real-time digital software FM synthesizer, which aside of implementation of the FM synthesis is also going to implement these basic methods for changing generated tone.

Keywords: digital synthesis, FM synthesis, music synthesizer

Seznam použitých zkratek a symbolů

DSP	–	Digital Signal Processing
FFT	–	Fast Fourier Transform
FIR	–	Finite Impulse Response
FM	–	Frequency Modulation
IIR	–	Infinite Impulse Response
LFO	–	Low Frequency Oscillator

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

Jedním z cílů této práce je popsat základní techniky, kterými lze generovat zvuk v hudebních syntetizérech. Tyto metody by mělo být možno použít ke generování zvuku v reálném čase na osobních počítačích. Což samozřejmě omezuje výběr těchto metod syntézy nebo případně rozsah (například počet sinusoid generovaných v aditivní syntéze) jejich použití. Principy některých metod syntézy, které jsou zde popsány (aditivní a FM syntéza), jsou použitelné ne jenom pro digitální syntezátory, ale i pro syntezátory analogové. Avšak tento text se bude zabývat především syntézou digitální.

Dalším cílem této práce je popsat, jak lze měnit charakter tohoto vygenerovaného zvuku. K čemuž v syntezátorech existuje spousta modulů, které mohou například modulovat určité parametry generované vlny (např. frekvenci nebo amplitudu), vlnu filtrovat nebo přidávat efekty. Těchto modulů je v syntezátorech velké množství, které se u každého z nich liší. V tomto textu se objeví základní typy těchto úprav zvuku, které lze v podstatě nalézt skoro na každém moderním syntezátoru. Těmto metodám se věnují další části tohoto textu, které následují po popisu rozdělení syntezátorů a popisu metod syntézy. Opět je jejich výběr podřízen použitím v reálném čase.

A dalším cílem je tyto znalosti využít k implementaci digitálního softwarového FM syntezátoru, který bude implementovat jednak samotnou metodu FM syntézy, ale také bude implementovat i zde popsané způsoby změny vlastností tónu. Tento implementovaný syntezátor by měl pracovat v reálném čase na osobním počítači.

Tento text je psán ve snaze problematiku digitálních syntetizérů popsat srozumitelně pro člověka, který by chtěl takovýto syntetizér implementovat. Tématicky se digitální syntetizéry pojí k digitálnímu zpracování signálů (anglicky *DSP*, tj. *Digital Signal Processing*). Některé části syntezátoru jsou teoreticky náročné, například pokud se začneme bavit o digitálních filtrech. O návrhu digitálních filtrů jsou napsány celé knihy a jedná se asi o teoreticky nejsložitější část syntezátoru. Zaměřím se tedy především na minimální znalosti, které je vhodné mít, pokud chceme nějaký digitální filtr vybrat a implementovat.

2 Co je syntetizér

Syntéza označuje postup sjednocování nebo spojování, kdy sjednocením (syntézou) dvou a více entit vzniká entita nová. Používá se však i ve významu vytváření něčeho umělou cestou. Oba významy se dají ve vztahu k hudebním syntezátorům používat. Jednak můžou generovat signály naprosto umělé, a pak je třeba sjednocovat jednoduchým sčítáním (jeden signál je superponován na druhý) a nebo je vzájemně modulovat. Dále se slovo syntéza velmi často nachází v anglické literatuře o teorii DSP. Rovnice *zpětné Fourierovy transformace* se uvádí jako *Fourier transform synthesis equation* neboli „rovnice syntézy Fourierovy transformace“. (Na takové označení je možné narazit například v [8].)

Hudební syntezátor (nebo také syntetizátor/syntetizér) je elektronický hudební nástroj, který vytváří zvuk určitou metodou syntézy. Výsledný zvuk může být naprosto umělý a generovaný, může a nemusí připomínat nějaký skutečný hudební nástroj, nebo může přímo vzorky zvuku skutečného nástroje přehrávat. Jedná se většinou o samostatný přístroj, který může obsahovat i klaviaturu a jiné ovládací prvky, nebo může jít jen o samostatnou krabici, ke které se periferie externě připojují. Ačkoliv můžou už být dané periferie součástí samotného nástroje, tak komunikují se samotným syntezátorem přes MIDI protokol. Externí periferie se připojují přes MIDI rozhraní a komunikují rovněž přes MIDI protokol.

I levné elektronické klávesy používají nějaký typ syntézy. Ale na rozdíl od pravých syntezátorů neposkytují nastavitelné parametry, které by ovlivňovali přímo tvorbu zvuku. [9]

3 Rozdělení syntezátorů

Tyto základní způsoby rozdělení jsem čerpal z [9], ale samotná rozdělení jsou poměrně logická a zažitá. Hudební syntezátory se dají rozdělit několika způsoby. Například podle typu signálu se kterým pracují, podle typu syntézy nebo podle metody syntézy.

3.1 Podle typu signálu

Jedním ze základních rozdělení je dělení podle typu signálu se kterými pracují:

- analogové
- digitální
- hybridní

Čistě analogové syntezátory mají jak řídící obvody, tak i obvody pro generování a zpracování zvuku analogové. Parametry syntezátoru se tedy ovládají přes potenciometry, které ovlivňují napětím řízené oscilátory, ladí filtry, apod. Výhodou oproti digitálním syntezátorům je, že zde nejsou problémy s aliasingem[1]. Mezi nevýhody patří nutné dolad'ování nástroje kvůli změnám teplot, které ovlivňují parametry elektroniky (např. odpor) nebo i to, že zde není způsob, jak si uložit nastavení jednotlivý parametrů syntezátoru.

Digitální syntezátory mají řídící elektroniku digitální, tedy jde například ukládat nastavení parametrů. A hlavně taky zvuk je generován a upravován digitálně, tudíž nástroj není závislý na teplotách součástek a nemusí se tedy ladit. Samozřejmě digitální nástroje umožňují zařadit další digitální ovládací prvky, jako jsou displaye a podobně. Co lze však dělat digitálně je omezeno výkonem součástek, v dnešní době tedy výkonem DSP čipů. I přesto však digitální syntezátor předčí analogové téměř ve všem, protože zvládnou to co analogové a ještě více.

Hybridní syntetizéry jsou v podstatě analogové syntetizéry po stránce generování a zpracování signálů. Navíc však obsahují digitální řídící obvody, takže se dají nastavovat parametry přesně číselně a zbytek udělají DA převodníky a zesilovače, které ovládají generátory, filtry a podobně. Uživatel většinou už může ukládat svoje nastavení, ale stále se většinou musí nástroj dolad'ovat. Tyto hybridy se i dnes vyrábějí, kvůli požadavkům na „analogový zvuk" některých puristů. [9]

3.2 Podle typu syntézy

Jedná se o trochu starší způsob rozdělení, který hlavně bere ohled na to, jak se dá upravovat výsledná barva zvuku. Uvádím ho zde proto, že se stále dá najít v literatuře, která se syntetizéry zabývá. Dvě hlavní větve syntézy jsou:

[1] Zkreslení signálu, které vzniká, když se snažíme samplovat nebo generovat signál obsahující frekvence vyšší jak $\frac{1}{2}f$ vzorkovací frekvence.

- aditivní
- subtraktivní

Aditivní metody syntézy se spoléhají na vytváření zvuku čistě sčítáním více signálů. Jak složité jsou tyto signály, na tom tolik nezáleží, ale prakticky se používají sinusoidy a to z poměrně jednoduchého důvodu. Pomocí frekvence, amplitudy a fáze jednotlivých sinusoid ovlivňujeme frekvenční spektrum zvuku. Teoreticky se takto dá poskládat jakýkoliv zvuk. Vzhledem k tomu, že jediným zástupcem čistě aditivního typu syntézy je aditivní syntéza, nechám další popis příslušné části textu.

Subtraktivní typ syntézy naopak pracuje už s nějakým složitějším, například vygenerovaným, signálem a barvu zvuku upravuje tak, že nějaké složky frekvenčního spektra původního signálu zeslabuje nebo úplně potlačuje („odčítá“ od původního signálu). To samozřejmě provádí pomocí filtrů. Nezáleží zde jakým způsobem byl původní signál vytvořen, ale měl by být bohatý na harmonické složky, aby bylo co filtrovat. Teoreticky se tu dá zařadit jakýkoliv syntezátor, který má ve svém řetězci filtr. Ať už používá klasickou subtraktivní syntézu (ta filtruje vlny základních tvarů jako pila, obdélní, atd.), tak i například wavetable syntézu, sample playback syntézu, granulární syntézu a další.

Zde se vlastně ukazuje zastaralost tohoto rozdělení. Toto rozdělení vzniklo v počátcích syntezátorů, kdy se používala aditivní syntéza nebo subtraktivní syntéza a přežilo až do dob modernějších syntezátorů, kdy ale začalo ztrácet smysl. Například některé FM syntezátory se zde těžko daly zařadit. Například FM syntezátor Yamaha DX7 neměl žádný filtr, místo toho se dala modulovat (například obálkou nebo oscilátorem) hloubka FM modulace, což do jisté míry vytvářelo podobný efekt. Zároveň však FM syntéza není aditivní syntézou, protože se signály zde nesčítají, ale navzájem modulují. Zároveň ale nic nebrání filtr do FM syntezátoru přidat. Některé FM syntezátory tedy mají i filtr, což zase vnáší o trochu více zmatku do tohoto relativně zastaralého rozdělení. [9]

3.3 Podle metody syntézy

Dále se dají syntezátory rozdělit podle samotné metody syntézy:

- aditivní
- subtraktivní
- FM syntéza
- PM syntéza
- granulární syntéza
- wavetable syntéza
- sample playback syntéza
- atd.

Metoda syntézy se vztahuje především ke generátorům tónu. Některé tyto metody si dále popíšeme.

3.3.1 Aditivní syntéza

V této části textu věnované aditivní syntéze jsem čerpal z [5], [9], [4] a [7]. Fourierova řada a transformace naráží na velice zajímavý koncept. Trochu zjednodušeně řečeno, každý zvuk v přírodě se dá rozložit na řadu/sérii sinusových a kosinusových složek o různé amplitudě a frekvenci. Nebo-li také na sérii sinusových složek o různé amplitudě, frekvenci a fázi. Teoreticky bychom mohli analyzovat například tón nějakého hudebního nástroje, třeba klavíru, a následně ho reprodukovat pomocí umělých oscilátorů (ať digitálně nebo analogově). Tón bychom nemuseli jen reprodukovat, ale i upravovat jeho výšku a dynamiku, a třeba simulovat útlum jednotlivých harmonických složek[2] podle síly úhozu klávesy. Tak bychom mohli dosáhnout poměrně věrné simulace klavíru. Avšak tato série/řada může být nekonečná, aby dokázala dostatečně aproximovat původní složitý signál. Což je v digitálním zpracování dat poměrně obsáhlý problém (prostě nelze generovat nekonečný počet sinusoid).

Aditivní syntéza tuto ideu značně zjednodušuje a většinou se omezuje na vytváření signálů periodických. Často využívá spíše jen sinusové generátory a dokonce zanedbává i fázové posuny. Také nemůžeme generovat nekonečně mnoho sinusových signálů. Počet vygenerovaných sinusovek je tedy například omezen výpočetním výkonem, kde samozřejmě ještě hraje roli, jestli půjde o real-time syntézu, nebo si můžeme na výsledek počkat. Frekvence jednotlivých složek jsou celočíselnými násobky základní frekvence. Těmto složkám s vyšší frekvencí od základního tónu se také říká vyšší harmonické frekvence, v hudební teorii také alikvótní tóny. Výsledný tón tedy i ladí/nezní falešně.

Takový zjednodušený výpočet aditivní syntézy může být vyjádřen třeba takto [5]:

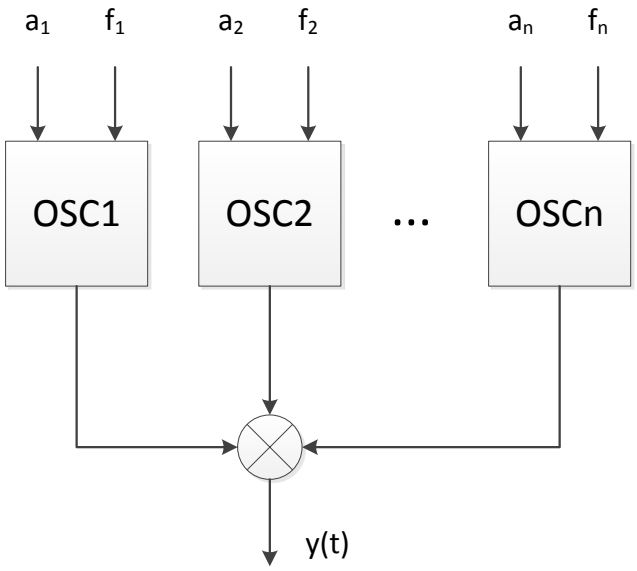
$$f(t) = \sum_{k=1}^{n} a_k \sin(2\pi f_k t) \tag{1}$$

- a_k je amplituda k-té sinusové složky
- f_k je frekvence k-té sinusové složky
- n je počet sinusových složek
- t je čas

Blokové schéma takového imaginárního stroje nebo programu je pro ilustraci znázorněno na obrázku č. 1. Schéma přibližně vychází ze vzorce č.1. Všechny tyto sinusové složky přispívají k harmonicky bohatému zvuku. Jak již bylo zmíněno výše, tak počet těchto sinusovek je v praxi značně omezený, a tak prakticky nemůžeme třeba rekonstruovat tón klavíru. Samozřejmě čím více sinusových a kosinusových složek původní signál aproximuje, tím lépe bude tón znít a více se podobat originálu.

Aditivní syntéza se tedy většinou nepoužívá k simulaci nástrojů, ale ke generování nějakých jednodušších tónů. Často se nepoužívá v takto čisté formě, ale v kombinaci

[2] *Harmonické složky signálu* určují frekvenční spektrum signálu. Jedná se o sinusoidy, jejichž frekvence jsou celočíselnými násobky základní (nejnižší) frekvence obsažené v signálu. Říká se jim také *vyšší harmonické*.



Obrázek 1: Ilustrační příklad aditivní syntézy.

s nějakými jinými technikami, například Wavetable syntézou [5]. Wavetable syntéza, jak název napovídá, má uložené nějaké periodické zvukové vlny do paměti (většinou jednu periodu každé vlny), a ty pak přehrává nakombinované dohromady v určitých poměrech nebo samostatně. Můžeme proto například aditivní syntézou předem vygenerovat a uložit do tabulek v paměti určité složitější zvukové vlny nebo jejich části. A ty poté přehrávat a kombinovat/sčítat. V tabulce by měla být uložena jedna perioda signálu, tedy pro prodloužení tónu přehrajeme danou periodu vícekrát. Můžeme i měnit výšku pomocí „rychlejšího“ přehrávání periody. To například tak, že nebudeme zvětšovat index (přes který se prochází tabulka) po jedné, ale po dvou. Tím se i přečte celá perioda vlny za poloviční čas. Frekvence se tedy zvýší dvakrát. Další celočíselné inkrementy fungují stejně. Problém nastane, pokud chceme frekvenci zvýšit o nějaký neceločíselný násobek. V takovém případě musíme použít nějaký typ interpolace, třeba lineární interpolaci. Výsledek potom není samozřejmě přesný a vznikají různá i slyšitelná zkreslení. Nejedná se tedy o ideální řešení, ale přesto se používá.

Aditivní syntéza taky umožňuje generovat základní typy signálu, jako je například pila, obdélník a trojúhelník. Nelze takto vytvořit třeba dokonalý obdélníkový signál, pro ten by zase teoreticky bylo potřeba nekonečně mnoho sinusových složek. Ale můžeme takto tyto signály alespoň aproximovat. Navíc se v digitální syntéze poměrně hodí omezovat maximální frekvenci sinusových složek signálu pod polovinu vzorkovací frekvence. Tak dodržíme Nyquistův teorém a předejdeme aliasingu.

Tvar	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
Sinus	1	0	0	0	0	0	0	0	0
Obdélník	1	0	$1/3$	0	$1/5$	0	$1/7$	0	$1/9$
Trojúhelník	1	0	$-1/3^2$	0	$1/5^2$	0	$-1/7^2$	0	$1/9^2$
Pila	1	$1/2$	$1/3$	$1/4$	$1/5$	$1/6$	$1/7$	$1/8$	$1/9$

Tabulka 1: Příklad amplitud sinusových složek signálu pro základní typy vln

Všechny tyto základní vlny se skládají z vyšších harmonických, které mají frekvenci celočíselného násobku základní frekvence. Příklad prvních 9 hodnot amplitudy pro dané harmonické složky těchto základní tvarů je v tabulce č. 1 [10]. Obdélníková vlna obsahuje jen liché harmonické složky a amplituda klesá s rostoucí frekvencí tak, že pro každou x-tou harmonickou je amplituda na $1/x$ hodnoty amplitudy a_1 základní frekvence. Trojúhelníková vlna má pouze liché harmonické složky a amplituda klesá s každou x-tou harmonickou na $1/x^2$ hodnoty a_1 a zároveň mění střídavě znaménko. Pila má všechny harmonické složky a jejich amplituda klesá na $1/x$ hodnoty a_1 pro x-tou harmonickou.

Aditivní syntéza v její čisté formě prakticky nepotřebuje filtr, protože frekvenční spektrum signálu přímo ovlivňujeme tím, jaké frekvence sinusovek generujeme a jejich amplitudou.

Poznámka k implementaci

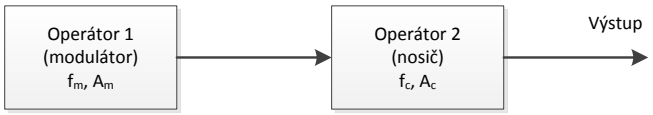
Nutno podotknout, že vzorec č.1 obsahuje proměnou času, zatímco například program si čas musí vypočíst pomocí vzorkovací frekvence a aktuálního vzorku, a hodnotu sinusovky spočítat pro každý vzorek zvlášť. Většinou se ale používá akumulátor fáze, kde se přičítá inkrement fáze, vypočtený právě ze vzorkovací frekvence. Tedy například takto vypočteme inkrement [5]:

$$inkrement = \frac{2\pi}{f_{vzor}} f_{osc}$$

Ten stačí vypočíst jednou, nebo při každé změně frekvence oscilátoru f_{osc}. Vzorkovací frekvence f_{vzor} se většinou v průběhu běhu programu nemění. Tento inkrement přičítáme do akumulátoru fáze, tedy takto:

$$aktualni\_faze = aktualni\_faze + inkrement$$

A akumulátor fáze už přímo použijeme k výpočtu hodnoty sinu, tedy $a_{osc} \sin(aktualni\_faze)$, kde a_{osc} je amplituda oscilátoru. Inkrementace akumulátoru fáze se provádí pro každý další vzorek, který je třeba vypočíst. Dobré je držet hodnoty akumulátoru v intervalu $[0, 2\pi]$. To provedeme jednoduše přes podmínku, kdy budeme odčítat hodnotu 2π od akumulátoru fáze, jestliže hodnota akumulátoru je větší jak 2π. Tento příklad jen ilustruje, jak toto provést pro jednu sinusovku. Pro sumu sinusovek je třeba mít akumulátory fáze pro každý oscilátor a provádět celý proces v cyklu. Jinak tato technika použití akumulátoru fáze se dá v programu použít pro jakýkoliv oscilátor, ne nutně jen sinusový.



Obrázek 2: Jednoduchá FM syntéza

3.3.2 FM syntéza

V této části textu věnované FM syntéze jsem čerpal z [2], [4], [5] a pro českou terminologii používám názvy z [9]. Tento způsob syntézy náhodou objevil John Chowning, který tehdy byl zaměstnancem Stanfordské univerzity. V té době se už běžně používaly nízkofrekvenční generátory k modulaci výšky (tedy frekvence) generovaného tónu. V případě, že jeden sinusový generátor, který moduluje frekvenci výstupního (např. sinusového) signálu, pracuje na frekvencích pod slyšitelnou hranicí (tedy pod 20 Hz), tak výsledným efektem je vibráto. Pokud ale frekvence modulátoru překročí tuto hranici a dostane se frekvencí do slyšitelného spektra, tak se vibráto vytrácí a výstupní signál místo toho mění barvu tónu.

Tento princip ke generování frekvenčně bohatého signálu byl patentován a licence prodána firmě Yamaha. Patent už ale před nějakou dobou vypršel, takže je dnes bezpečné tento princip i komerčně využívat [5]. Yamaha v době používání tohoto patentu vyvíjela ve spolupráci s Chowningem komerční digitální FM syntetizér. Jedním z prvních tedy byl Yamaha GS1, který ale byl považován za velmi drahý. Mnohem více se rozšířil jejich pozdější model Yamaha DX7. Model DX7 následovali i další dražší nebo levnější verze, lišící se počtem hlasů, počtem navzájem modulovatelných oscilátorů a algoritmů.

Teď něco málo k názvosloví. *Operátor* je jakýkoliv oscilátor podílející se na FM syntéze, bez ohledu zda se jedná o oscilátor nosné nebo modulační vlny. Oscilátory jejichž výstupy jdou slyšet se nazývají *nosiče*. Zatímco ty co slyšet nejdou přímo, ale podílejí se na modulaci nosičů, se nazývají *modulátory*. *Algoritmus* v kontextu s FM syntézou je schéma, které popisuje zapojení jednotlivých operátorů (modulátorů a nosičů).

Výsledný zvuk ovlivňujeme poměry frekvencí mezi modulátory a nosiči, a také hodnotami jejich amplitud. Například nejjednodušší možná kombinace operátorů je jeden modulátor, který moduluje frekvenci jednoho nosiče. Toto je znázorněno v zjednodušeném blokovém schématu na obrázku č. 2, kde Operátor 1 moduluje sinusovým signálem o frekvenci f_m a amplitudě A_m Operátor 2, který pracuje na frekvenci f_c s amplitudou A_c. Jelikož Operátor 2 je zároveň výstupem, který půjde ve výsledku slyšet, tak jeho amplituda mění hlasitost a jeho frekvence výšku tónu. Vzorec pro výpočet takovéto FM modulace může vypadat takto [5]:

$$e = A_c \sin(2\pi t(f_c + A_m \cdot \sin(2\pi f_m t))) \tag{2}$$

- e je aktuální hodnota amplitudy nosiče
- A_c je amplituda nosiče
- f_c je frekvence nosiče
- A_m je amplituda modulátoru
- f_m je frekvence modulátoru
- t je čas

Ve vzorci č.2, A_m určuje maximální variaci frekvence f_c nosiče. Frekvence modulátoru f_m udává, jak rychle se mění frekvence nosiče. Než přímé zadávání frekvence modulátoru, se častěji zadává tzv. *c/m ratio* (v angličtině). Což je poměr mezi frekvencí nosné (c jako carrier) a frekvencí modulátoru. Toto je důležité, protože pokud je tento poměr racionální číslo, tak výsledkem syntézy bude signál, jehož spektrum je harmonické[3]. Tedy pro signál s harmonickým spektrem je třeba, aby poměr c/m byl $c/m = N_1/N_2$, kde N_1 a N_2 jsou přirozená čísla. Takový tón ve své podstatě ladí. Pokud poměr c/m je iracionální číslo, jako třeba $1/\sqrt{2}$, tak vzniká signál s neharmonickým[4] spektrem. Což prakticky znamená, že tón se bude zdát falešný, protože jeho části spektra nebudou ladit.

Dále je tu menší problém s konstantní amplitudou A_m modulátoru. Amplituda A_m určuje, jak moc se bude pohybovat výsledná frekvence nosné od své základní frekvence f_c. Mějme frekvenci nosiče f_c rovnu 200 Hz. Pokud se A_m rovná 100, potom způsobuje maximální výchylku 100 Hz na obě strany (plus i mínus) od frekvence nosiče. Maximální frekvence obsažená ve výsledném signálu tedy bude 300 Hz a minimální 100 Hz. Což je změna frekvence f_c o 50 %. Pokud ale bude frekvence f_c rovna 1000 Hz, tak změna frekvence bude pouze 10 %. Což znamená, že barva tónu se bude značně měnit s frekvencí.

Proto se většinou při FM syntéze nenastavuje amplituda A_m modulátoru přímo, ale zase v nějakém poměru. Tomuto poměru se říká *modulační index*, většinou se označuje písmenem I a je dán vztahem [2][5]:

$$I = \frac{\Delta f_c}{f_m} \tag{3}$$

Kde Δf_c je maximální deviace (odchylka) frekvence signálu nosné vlny a f_m je frekvence modulátoru. Tento index nám tedy udává maximální změnu nosné frekvence poměrově. A proto tento poměr také říká, čemu se bude rovnat A_m. V podstatě Δf_c odpovídá hodnotě A_c, tu tedy vypočteme $A_c = I f_m$. Pokud by tedy například bylo $I = 1/10$, potom by byla maximální změna frekvence v rozsahu celého nástroje všude 10 %, a proto barva tónu by zůstala stejná pro všechny frekvence.

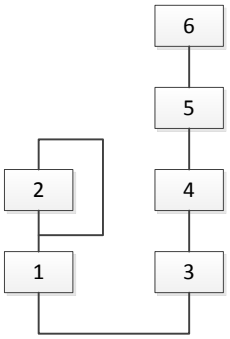
V FM syntéze se používají i hodnoty I větší než jedna, což může vést až ke generování záporných frekvencí nebo překročení poloviny vzorkovací frekvence. S oběma případy

[3]Harmonické spektrum je takové, jehož vyšší harmonické frekvence jsou celočíselným násobkem frekvence základní(nejnižší).

[4]Neharmonické spektrum je takové, jehož vyšší frekvenční složky nemají frekvenci celočíselného násobku frekvence základní.

se vlastně v FM syntéze počítá. Záporné frekvence v podstatě jen přehodí fázi vlny (což člověk neslyší). Pro frekvence vyšší než je polovina vzorkovací frekvence dochází k aliasingu, ale ten se projevuje tak, že se tyto frekvence „odráží“ zpátky do použitelného spektra (frekvence nižší než polovina vzorkovací frekvence) a zvyšují nebo snižují amplitudy už vygenerovaných harmonických. Toto zvyšování a snižování amplitud spektra se děje relativně pseudonáhodně v závislosti na c/m poměru a hodnotě I. Důležité je, že pro racionální c/m poměry bude i pro $I > 1$ generovaný signál mít harmonické spektrum.

Už z matematiky, která se v FM syntéze odehrává, je jasné, že tvorba zvuku touto metodou není příliš intuitivní. Jen těžko se tedy dá používat například k napodobování skutečných nástrojů. Celý příklad (obrázek 2 a výpočty parametrů) uvedený zde je navíc pro jednoduchou FM syntézu s dvěma operátory. V FM syntéze ale může být operátorů více a můžou být různě propojeny. Například z komerčních produktů softwarový syntetizér FM8 od Native Instruments může využít až 8 operátorů, které lze libovolně zapojit. Tyto algoritmy bývají například graficky značené jednoduchými bloky s čísly operátorů a čárami propojujícími vstupy a výstupy. Někdy tedy podobně jako na obrázky č.2.



Obrázek 3: Příklad složitějšího algoritmu FM syntézy

Složitější příklad algoritmu FM syntézy je na obrázku č.3. Tento způsob značení se dá najít například pro syntezátor Yamaha DX7, ale i jinde. Čte se poměrně jednoduše od shora dolů. Tedy operátor č.6 moduluje operátor č.5, ten zase moduluje operátor č.4, a tak dále. Zajímavá je zpětná vazba operátoru č.2, občas se v FM syntéze používá i takováto sebe modulace. Operátory 1 a 3 jsou tedy nosiči a jejich výstup se sčítá. Upravovat zvuk takovéhoto složitého algoritmu, je samozřejmě poměrně obtížné. Hlavně kvůli spoustě vazeb, úprava parametrů jednoho operátoru se značně promítne do výsledného zvuku. Podobně složité algoritmy se dali najít i na DX7, ale jejich operátory už měli přiřazené výchozí hodnoty parametrů od výrobce, které odpovídali nějakému nastavení pro určitý zvuk (třeba „Elektronické piáno“). Z čehož už se poté snadněji vychází, pokud chceme

zvuk takového algoritmu měnit. Tvorba zvuku je tedy hlavně o experimentování s parametry.

Co se programové implementace algoritmů s více operátory týče, tak se v podstatě řešení neliší od vzorce č.2. Pokud jsou operátory sériově za sebou (např. operátory 6, 5 a 4 na obrázku 3), tak se prostě výsledek jedné FM modulace použije pro další následující modulaci v sérii. V případě zpětné vazby si musíme pamatovat předchozí vypočtený vzorek. A výsledky více nosičů se jednoduše sčítají.

3.3.3 Sample-playback syntéza

V této části textu věnované sample-playback syntéze jsem čerpal z [1], [6] a [9]. Sample-playback syntéza, nazývána také „Sample based" syntéza, je jedna z nejrozšířenějších typů syntézy signálu. Existuje v hardwarových i softwarových syntezátorech a jedná se o digitální syntézu. Je založená na principu přehrávání vzorků signálů, které jsou předem předem uloženy v paměti. V tomto ohledu se podobá wavetable syntéze. Ale na rozdíl od wavetable syntézy nemusí obsahovat jen jeden cyklus vlny. Vzorky sample-playback syntézy můžou obsahovat i neperiodické signály zvuku, jako například zvuky bubnů.

V podstatě nejsou kladena žádná omezení na zdroj těchto uložených zvuků. Mohou být uměle generovány jinými metodami syntézy, jako je například aditivní syntéza nebo FM syntéza. Ale mohou to být i zvuky nahrané z reality. V tomto ohledu poskytuje sample-playback syntéza relativně nenáročnou simulaci hudebních nástrojů, ve srovnání například s fyzikálním modelováním nástrojů.

Kvalita generovaného zvuku přímo závisí na kvalitě nahraných vzorků zvukových signálů. A jejich kvalita reprodukce je omezena vzorkovací frekvencí a bitovou hloubkou, s kterou byli nahrány. Oba parametry jsou samozřejmě omezovány výpočetní silou stroje, který syntézu provádí.

Výhoda sample-playback syntézy je tedy možnost simulovat jiné hudební nástroje nebo akustické jevy. Je možno vzorkovat i uměle generované vlny. Lze tedy například vzorkovat základní typy signálů, jako je sinusovka, pila, obdélník a trojúhelník. A tak může syntetizér napodobovat zvuk starých subtraktivních syntezátorů. Nevýhodou je samozřejmě omezená manipulace se signály uloženými v paměti. Například nemůžeme měnit střídu[5] obdélníkového signálu nebo sklon pily.

Jedním z hlavních způsobů, jak již uložený signál dále upravovat, je filtrovat jej. Sample-playback syntezátory proto většinou implementují celou řadu filtrů (horní propusti, dolní propusti, pásmové propusti a zádrže). Proto se občas tato metoda syntézy řadí mezí třídu subtraktivních syntéz (staré, ale pořád používané rozdělení). V kontextu tohoto typu syntézy se často pojmem „vzorek" označuje i celý uložený tón/zvuk. Což se termínově trochu plete se vzorkem zvuku navzorkovaného (digitálního/číslicového) signálu.

[5]Střída obdélníkového signálu je poměr, který udává dobu trvání jeho maximální amplitudy k době trvání minimální amplitudy v rámci jedné periody. Udává se buď poměrem (jako například 1:1, 2:1, atd.) nebo procentuálně. Střída 1:1 odpovídá hodnotě 50 %, tedy signál bude polovinu periody na maximální hodnotě a polovinu periody na minimální hodnotě.

Změna výšky tónu

Často je potřeba měnit výšku tónu. Což se standardně provádí rychlejším nebo pomalejším přehráním uloženého signálu. V hardwarových syntezátorech (hlavně těch starých) se tohle dalo provádět například ovládáním pracovní frekvence digitálně analogového převodníku. Běžně se ale provádí tzv. *převzorkování* signálu, kdy už digitalizovaný signál znova vzorkujeme s jinou frekvencí vzorkování.

To se prakticky provádí tak, že si vytvoříme čítač/ukazatel, který drží pozici aktuálního přehrávaného vzorku signálu. Pokud by byl tento ukazatel jen celočíselný, tak by mohl ukazovat jen na přesně jeden vzorek signálu. To by nám značně limitovalo výšky tónů, které bychom mohli takto vytvořit (vysvětlení dále). Vhodnější tedy je nějaká proměnná, která udrží číslo s plovoucí čárkou. Chceme tedy například, aby přehraný signál měl 2 krát vyšší frekvenci oproti uloženému originálu. (Tedy chceme tón o oktávu vyšší.) To znamená, že vlastně chceme, aby perioda signálu byla poloviční. Budeme proto inkrementovat náš ukazatel číslem 2, a takto bez problému přečteme celý uložený zvuk. Pokaždé bude totiž ukazatel ukazovat přímo na vzorek originálního signálu. Pro další celočíselné násobky frekvence se toto provádí stejně. Například pro 3 krát vyšší frekvenci oproti originálu budeme inkrementovat ukazatel po třech.

Problém nastane, pokud chceme zvýšit frekvenci o nějaký neceločíselný násobek. Jedním řešením je ignorovat vše za desetinou čárkou a prostě vzít jen jeho celou část k určení vzorku, který se má přečíst. To je ale hodně nepřesné řešení, při kterém vznikají různá zkreslení a šum. O něco lepší je zaokrouhlovat, ale ani to nám neposkytne dobrou kvalitu zvuku.

Mnohem lepším řešením je použít nějakou interpolaci. Asi nejpoužívanější interpolací je lineární interpolace. Ta v podstatě počítá jakýsi vážený průměr dvou nejbližších hodnot, mezi které ukazatel zapadá. Aproximuje hodnotu mezi dvěma body rovnou čarou. Ani toto není úplně ideální řešení, ale používá se. Pak je ještě možné používat interpolace vyšších řádů, tedy nějaké křivky, které jsou proložené více body.

V každém případě, interpolace vždy jen aproximuje výsledek. Skutečnou hodnotu, například analogového signálu, která nebyla navzorkovaná, prostě už získat nejde. Podobně funguje i snižování frekvence tónu, jen inkrementujeme ukazatel o hodnotu menší jak 1. Dá se říct, že tady už je nějaká interpolace nutná. Nějaké vícenásobné opakování jedné hodnoty nepřipadá moc v úvahu. [6]

Looping

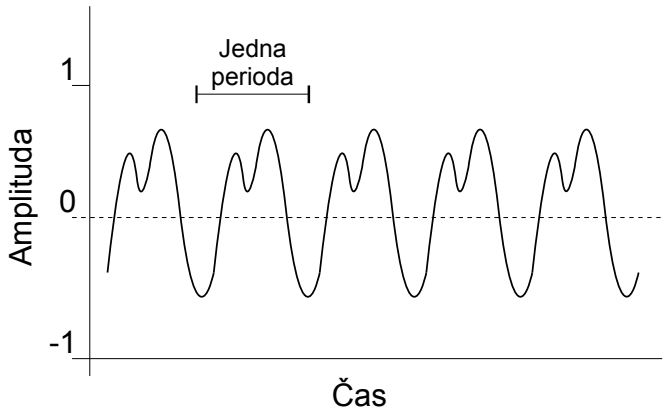
S přidržením klávesy klaviatury syntetizéru často očekáváme, že daný tón bude znít po celou dobu držení klávesy nebo minimálně jeho délku můžeme ovlivňovat. V sample-playback syntéze jsou navzorkované signály nějaké určité délky uloženy v paměti. A my potřebujeme tyto signály prodloužit nejlépe tak, aby se nezměnila výška tónu.

Looping je metoda, která toto zajišťuje. Jak již název napovídá, dělá to tak, že přehrává nějakou část signálu pořád dokola. Tóny skutečných nástrojů se dají rozkouskovat na několik fází. Například v navzorkovaném tónu piána lze nalézt část, která odpovídá úhozu paličky do struny, kdy jde trochu slyšet i úhoz paličky, rychlý nárůst hlasitosti tónu

a mírné zvýšení výšky tónu. Poté tón zní po dobu držení klávesy s postupným klesáním hlasitosti. Poté například klávesu pustíme a tón se utlumí.

Je tedy třeba určit, která část signálu se bude ve smyčce přehrávat. Určení začátku a konce smyčky (v angličtině *looping points* [6]) je tedy klíčové pro výsledný zvuk znějícího tónu. Vybíráme kus části mezi počáteční a konečnou fází tónu, tedy část z fáze kdy tón zní.

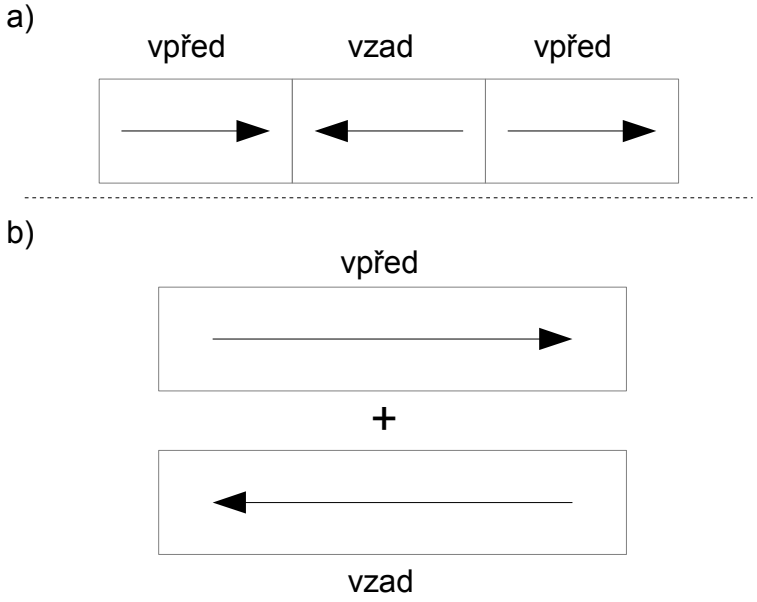
Existují algoritmy detekce výšky tónu (v angličtině *pitch detection* algoritmy), které hledají v signálu nejkratší periodu, která se v signálu opakuje. Zjednodušený příklad je na obrázku č.4, ale obecně jsou signály skutečných nástrojů ještě mnohem složitější. Perioda by se tedy neopakovala přesně takto stejně, ale spíš by se části signálu jen více či méně podobaly.



Obrázek 4: Příklad periody složitějšího signálu

Pokud ale použijeme pouze jednu periodu, jako na obrázku č.4, tak výsledný tón bude znít velmi uměle. Zvuk může ztratit svůj charakter a už nebude třeba možné poznat hudební nástroje, jemuž původně tón patřil. Proto se vybírá většinou větší část signálu, která obsahuje více těchto podobných period. (Samozřejmě někdo může účelně vytvářet uměle znějící tóny z reálných nahrávek.)

Máme tedy vybranou část signálu, která se bude opakovaně přehrávat. Bohužel ne vždy začátek a konec této smyčky na sebe dokonale navazuje. Pokud nejde například mírným posunutí těchto konců pěkně signál navázat, tak vznikají strmé skoky mezi vzorky signálu, které vytvářejí zvuk klikání. Což je velmi rušivý element, když vezmeme v úvahu, že toto kliknutí se ozývá při každé iteraci smyčky. Našim cílem samozřejmě je vytvořit přechod konce smyčky zpátky na začátek co nejméně znatelný.



Obrázek 5: Znázornění obousměrného čtení signálu smyčky v loopingu

To lze provést například zahlazením obou konců smyčky nějakou amplitudovou obálkou, která amplitudu vlny postupně snižuje k nule na obou koncích. (Neplést s ADSR obálkou a jejího využití k modulaci amplitudy. Tato obálka je pouze pro zahlazení cyklu.) Tedy zajišťuje, aby náběh a konec vybraného signálu začínal z nulové amplitudy. Tato obálka může mít podobu přímky a amplitudu snižovat lineárně. Nebo to může být nějaká složitější křivka, která přechod konce zahladí více přirozeně.

Další možností je například přehrávat smyčku ve dvou směrech, tedy tam i zpátky (v angličtině *bidirectional looping*). To je znázorněno na Obrázku 5a. Nebo může být signál čtený oběma směry vrstven na sebe (znázorněno na Obrázku 5b), což také zamaskuje strmý přechod signálu na obou stranách.

Existují i složitější metody, které jsou postavené na analýze signálu. Například lze analyzovat zvuk smyčky (třeba přes *FFT*), náhodně trochu změnit fáze spektrálních komponent (sinusoid), a zpětně signál zase syntetizovat (zpětná *FFT*). A část se změnou fází opakovat s každou iterací smyčky, což vytvoří zvuk s méně opakujícím se charakterem. [6],[9]

Multisampling

Při použití sample-playback syntézy k simulaci skutečných hudebních nástrojů, zde nastává problém, že zvuk mění barvu (mění se tedy i zastoupení a rozestupy sinusoid ve frekvenčním spektru ve srovnání se základním tónem) s hlasitostí hry a výškou tónu. Proto například k dobré simulaci piána, je třeba vzorků spousta. Jednak pro různou sílu

úhozu klávesy, tak i pro různě vysoké tóny. Tato technika se nazývá *multisampling* a je velice rozšířená v moderních zástupcích tohoto druhu syntézy.

Proto například s jedním vzorkem tónu piána si nevystačíme k dobré simulaci nástroje. Při změnách výšky tónu technikami popsanými výše a jeho hlasitosti pouze změnou amplitudy signálu, se tón bude zdát čím dál více umělý, čím více se bude výškou a hlasitostí odchylovat od původního vzorkovaného signálu.

Tvorba takové sady vzorků samozřejmě není jednoduchá. Jsou firmy a jednotlivci, kteří se samplováním nástrojů živí, a tyto balíky předpřipravených vzorků prodávají. Elektronická piána jsou do jistých cenových relací také založena na této metodě syntézy. A často kvalita jejich zvuku je právě přímo úměrná bohatosti sady samplů. (Drahá elektronická piána používají už techniky pojící se s frekvenční analýzou a fyzikálním modelováním nástrojů.) [1]

3.4 Speciální kategorie

Zde spadají kategorie syntetizérů, které často používají typy syntéz popsaných výše. Ale nějakým způsobem se dostatečně odlišují, aby pro ně vznikl nějaký příhodný název kategorie.

3.4.1 SoftSynth

*Softsynth*y neboli softwarové syntezátory bychom mohli i vyčlenit do speciální kategorie. Je jasné, že se jedná o syntezátory digitální, avšak nejedná se o samostatný přístroj, ale o software na osobním počítači, který nahrazuje všechny funkce hardwarového syntezátoru softwarově včetně uživatelské rozhraní. I tyto syntezátory používají různé metody syntézy zvuku, a proto se dají dělit podobně jako hardwarové digitální syntezátory.

Vlastně softwarové syntezátory mají hodně blízko k jejich hardwarovým variantám. V minulosti byli klasické digitální syntezátory implementovány pomocí samostatných obvodů, ale časem se s vývojem polovodičové elektroniky čím dál více částí integrovalo do jedné součástky. V dnešní době většina digitálních syntezátorů běží kompletně na jednom DSP procesoru, který má v sobě nahraný program, jenž obsluhuje kompletně celý nástroj včetně samotné syntézy zvuku a efektů.

Je zde ovšem pár rozdílů. Kromě optimalizace DSP čipů pro zpracování digitálních signálů je na osobním počítači operační systém. Většina operačních systémů a v podstatě všechny operační systémy pro běžné uživatele nejsou real-time OS. V praxi to znamená, že nelze odhadnout, jak přesně dlouho se bude daný kód vykonávat. Toto se samozřejmě promítne i v odezvě syntezátoru, a tak například syntezátor nehraje okamžitě po stisku klávesy a vlastně ani nehraje s určitým známým zpožděním. [5],[9]

3.4.2 Virtual Analog

Dalším speciálním druhem syntezátorů jsou *Virtual Analog* syntezátory. Ty se snaží o více či méně věrohodnou reprodukci zvuku starých analogových syntezátorů. Poptávka o produkty tohoto typu stoupla v době, kdy spousta dnes již klasických syntezátorů z dob dřívějších se už dlouho nevyráběla a ceny použitých nástrojů byly mnohonásobně vyšší, než ve své době tyto nástroje stály. Dnes je tato skupina syntezátorů hojně zastoupena v softwarových syntezátorech, ale i výrobci syntezátorů například vyrábějí zvuková rozšíření pro své moderní syntezátory nebo i samostatné digitální verze svých dřívějších analogových modelů.

Jednou z možností jak přivést zvuk starých analogových syntetizérů do dnešní doby je nástroj samplovat. To nás ale potom velmi omezuje jen na pár samplovaných zvuků nástroje bez možnosti úpravy zvuku pomocí změny parametrů originálního syntezátoru. Nahraný zvuk můžeme filtrovat, přidat mu efekty, ale to nijak neovlivní, jak byl zvuk generován v originálním nástroji. Tato metoda se většinou nepovažuje za *Virtual Analog* syntézu.

Většinou se takový nástroj snaží simulovat/emulovat vnitřní elektroniku originálu pomocí DSP algoritmů. Obsahuje tedy všechny parametry, které se dali nastavovat u ori-

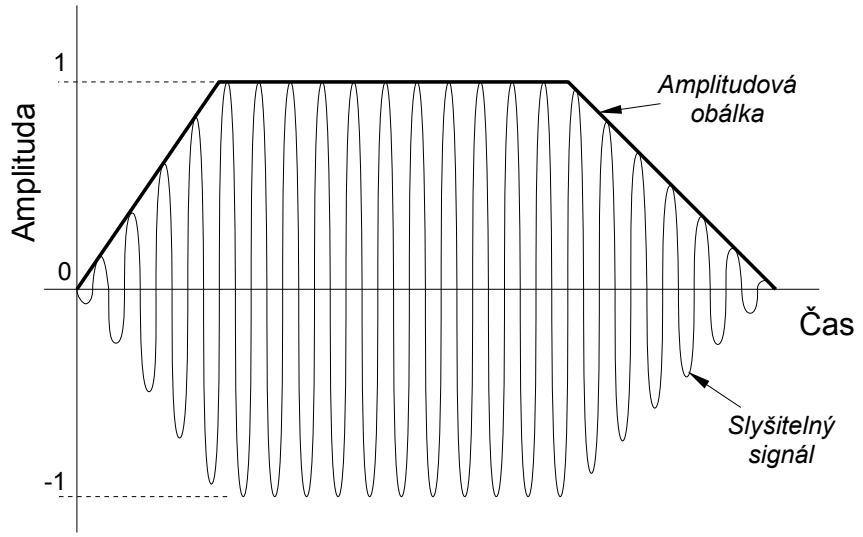
ginálního syntetizéru. Tyto syntetizéry mají i své výhody oproti originálům. Analogové syntezátory se musely například ladit, protože vlivem změny teplot mění elektronické součástky své elektrické vlastnosti. Oscilátory jsou především citlivé na tyto změny, což právě způsobovalo rozladění nástroje. Toto samozřejmě u digitálních nástrojů odpadá. Dalším častým omezením byla polyfonie, kdy staré analogové syntezátory byly například jednohlasé, nebo počet hlasů byl značně omezený. Jejich digitální verze mohou být polyfonní, nebo počet hlasů oproti originálu navyšovat. Často lze také ukládat nastavení nebo přidávat efekty. [9]

4 Generátory obálek

V této části textu věnované generátorům obálek jsem čerpal z [1], [5] a [9]. České názvosloví je použito z [9]. Skutečné hudební nástroje vytvářejí zvuk, který není po celou dobu svého trvání stejný. S časem se mění jeho barva, výška a hlasitost. Tón má svou náběhovou fázi, nějakou dobu zní, a pak zase zaniká. Uměle vytvořené generované tóny, například obdélník, můžou začínat okamžitě s maximální amplitudou, kterou si udrží po celou dobu svého znění, a pak okamžitě zaniknout s nulovou amplitudou. Přitom se ani nemusí změnit výška tónu nebo jeho barva. Takto vytvořený tón by byl ale velice nudný a zároveň by zněl velmi uměle.

Proto vznikly v syntezátorech tzv. *generátory obálek* (*envelope generators*), které se dají použít k modulování různých parametrů syntezátoru. Obálka tedy není signál slyšitelný ve výsledném zvuku, ale spíš řídícím signálem pro ostatní moduly (generátory, filtry, atd.) syntezátoru. (Dalším typem generátoru, který ovlivňuje jiné moduly řídícím signálem, jsou *nízkofrekvenční oscilátory*. O těch ale až později.)

Název *obálka* možná působí trochu zvláštně. Pokud modulujeme nějaký parametr, například výšku nebo amplitudu signálu, tak změny průběhu toho parametru v čase budou opisovat průběh obálky, který jej moduluje. Například modulací amplitudy výstupního signálu digitálního syntetizéru provedeme tak, že hodnoty signálu jednoduše násobíme modulačními hodnotami obálky. Obálka modulující amplitudu se nazývá *amplitudová obálka*, ilustrační příklad je na obrázku č.6.



Obrázek 6: Příklad amplitudové obálky

Na obrázku č.6 je amplitudová obálka vyznačena přímo ve stejném grafu jako i slyšitelný signál. Tím není samozřejmě myšleno, že by signál obálky byl obsažen ve slyšitelném signálu na výstupu. Tento příklad se jen snaží ilustrovat, jak modulovaná amplituda obálkou opisuje tvar této obálky. Modulovaný signál je řekněme nějaká sinusovka s kon-

stantní amplitudou a frekvencí po celou dobu svého trvání, jen možná na samotném obrázku to nejde úplně poznat, protože je nakreslen jen pro ilustraci. Způsob názvosloví použitý v názvu *amplitudové obálky* se dá aplikovat i na jiné obálky. Jako jsou například *výšková obálka* nebo *obálka filtru*, které samozřejmě modulují výšku tónu a mezní frekvenci filtru.

Signál obálky může být sdílen více moduly syntezátoru společně, a nebo každý modul může mít svůj vlastní generátor obálky. Generátory obálky jsou obvykle řízeny nějakým ovládacím prvkem syntezátoru, jako například klaviaturou nebo snímací membránou elektronických bubnů.

Signál obálky má většinou tvar takový, aby nějakým způsobem kopíroval fáze průběhu tónu (náběh, zánik, znění, atd.). A právě proto pomocí obálky můžeme lépe simulovat chování skutečných zvuků, nebo prostě umělým signálům přidat trochu přirozenější průběh. Jaké fáze obálka bude mít, je otázka designu syntetizéru. Typů obálek za léta vývoje vznikla spousta, ale asi nejstandardnější typem obálky je ADSR obálka (více v následující podkapitole).

Tento typ generátorů se nacházel už v analogových syntezátorech. Ale nachází se v podstatě v každém dnešním moderním syntezátoru. V dobách analogové syntézy se tedy o tento řídící signál staraly analogové generátory, jejichž obvody byli složeny například z integračních článků a napěťově ovládaných operačních zesilovačů (ty se dají využít i ke konstrukci komparátorů napětí, integračních článků, klopných obvodů a samozřejmě zesilovačů). Nemá sice cenu zabíhat do detailů, ale jen tak pro představu, nabíjením a vybíjením integračního článku s určitou integrační konstantou (určuje rychlost nabíjení) jde vytváře relativně lineárně rostoucí nebo klesající úseky napětí v čase. Intervaly nabíjení a vybíjení se dají měnit potenciometry, a tak i měnit sklon těchto úseků. Komparátory napětí, na které bylo přivedeno referenční napětí ovládané také potenciometry, zase mohli ve spolupráci s dalšími obvody přestat vybíjet/nabíjet integrační článek a držet tak konstantní hodnotu napětí. Samozřejmě čím složitější obálka, tím složitější byl tento generátor.

V digitálních hardwarových syntezátorech se pro tyto generátory zpočátku používali kombinace čítačů a vnitřní logiky sestavené z logických obvodů. Dnes je ale v podstatě vše software, protože moderní hardwarové syntezátory používají specializované DSP procesory. Program prakticky generuje vzorky, kterými „kreslí“ rovné čáry nebo případně křivky. Ty slouží k další modulaci jiných modulů syntezátoru, které dnes již také jsou jen moduly softwarovými.

4.1 ADSR obálka

V této části textu věnované ADSR obálce jsem čerpal z [5] a [9], české názvosloví použito z [9]. Především popis praktického využití čerpám z [9]. *ADSR obálka*, anglicky *ADSR envelope*, je jedním ze základních typů obálek. *ADSR* je zkratka slov *Attack*, *Decay*, *Sustain* a *Release*. Což jsou názvy nastavitelných parametrů ADSR obálky, které odpovídají stejnojmenným fázím tvořeného tónu. Následují stručný popis těchto fází.

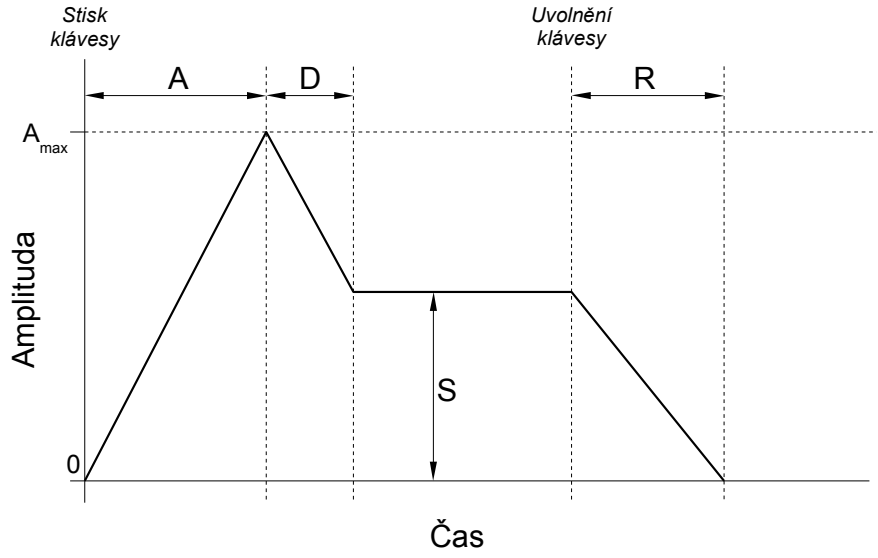
Attack Je úvodní náběhová fáze tónu, kdy amplituda vzrůstá (většinou strmě) k maximální amplitudě signálu.

Decay Je fáze, kdy amplituda klesá z maximální hodnoty na hodnotu *Sustain*.

Sustain Je fáze znění tónu. Tedy například pokud držíme klávesu klaviatury, tak tón po dobu držení klávesy zní.

Release Je fáze ukončování tónu, kdy tón z hodnoty amplitudy fáze *Sustain* klesá k nule. Tedy je to fáze následující například po puštění klávesy klaviatury.

Důležité je říci, co přesně tyto parametry označují, tedy co těmito parametry nastavujeme v syntezátoru. Jednotlivé části obálky jsou vyznačeny na obrázku č.7. Zde je vidět, že fáze *Attack*, *Decay* a *Release* jsou udávány časem, zatímco *Sustain* je udáván hodnotou amplitudy. Taky lze vidět, že amplituda této obálky roste nebo klesá lineárně, tedy přechody mezi fázemi jsou tvořeny v grafu přímkami.



Obrázek 7: ADSR obálka

Parametr *Attack* tedy prakticky udává, za jak dlouho po stisku klávesy přejde signál obálky z nulové amplitudy na hodnotu maximální amplitudy A_{max}. Hodnota amplitudy A_{max} bývá většinou nezměnitelná a v digitálních syntezátorech pracujících s čísly s desetinou čárkou většinou rovna jedné. Parametr *Decay* zase udává, za jak dlouho amplituda obálky klesne z maximální amplitudy A_{max} na hodnotu udávanou parametrem *Sustain*. *Sustain* je tedy hodnota amplitudy obálky v průběhu znění tónu. A jak je vidět z obrázku č.7, tak se v průběhu znění tónu nemění, je tedy konstantní. Parametr *Release* udává, za jak dlouho z úrovně *Sustain* klesne amplituda obálky k nule. Fáze *Release* začíná puštěním klávesy.

Na obrázku č.7 jsou parametry *Attack*, *Decay* a *Release* vztažené k času. Občas se ale udávají i pomocí poměrné, relativní, rychlosti (anglicky *Rate*), která v podstatě popisuje sklon nebo rychlost stoupání přímky, která tvoří průběh těchto částí obálky. Například ve směrnicovém tvaru rovnice přímky, kterou známe už ze základní školy, ve tvaru $y = kx + q$, by tento *Rate* parametr odpovídal směrnici přímky k. A například v digitálních syntezátorech by se dal *Rate* vyjádřit inkrementem za jeden vzorek signálu. Takovýto popis fází *Attack*, *Decay* a *Release* je pro nastavování trochu méně intuitivní než čas, ale v syntezátorech se používá také. (Tento odstavec v podstatě napovídá, jak se i řeší softwarová implementace ADSR obálky.)

Například zvuk piána ve své počáteční fázi má jakousi špičku v hlasitosti. Hlasitost prudce stoupá, a pak relativně rychle mírně klesne k hodnotě, která odpovídá fázi znění tónu. Což odpovídá přibližně fázím *Attack* a *Decay* ADSR obálky. Po uvolnění klávesy piána se tón utlumí, což zase neprobíhá okamžitě. Toto zase odpovídá fázi *Release* ADSR obálky. Fáze *Sustain* ADSR obálky už ale neaproximuje chování znějícího tónu piána velmi dobře. Tón piána ve fázi znění totiž neustále ztrácí energii a ztišuje se, což samozřejmě konstantní průběh *Sustain* fáze ADSR obálky nevystihuje. Proto například obálka použitá pro modulaci amplitudy simulace tónu piána by musela být trochu složitější. (Například přidat lineární útlum *Sustain* fázi.) ADSR obálka se ale dá použít v simulaci tónů jiných hudebních nástrojů, jako jsou například dechové nebo smyčcové nástroje, kde hráč může ovládat délku znění tónu. Nebo se ADSR obálky používají i k modulaci parametrů umělých tónů, aby zněli trochu přirozeněji a zajímavěji. Tento příklad s hlasitostí tónu odpovídá použití *amplitudové obálky*. Amplituda samozřejmě není jediný parametr, který se dá obálkami modulovat.

ADSR obálka se používá samozřejmě i k modulaci výšky/frekvence tónu. Například při počátečním „přefouknutí“ tónu dechových nástrojů se v úvodní části zvuku kromě amplitudové špičky ukazuje i mírné rozladění tónu. To se zase dá aproximovat v části *Sustain* a *Decay* ADSR obálky, kterou modulujeme výšku výsledného tónu.

Jelikož skutečné hudbení nástroje mění i barvu tónu v průběhu jeho znění, tak se nabízí použití ADSR obálky i k modulaci mezní frekvence filtru. Zase například piáno s větší silou hry tvoří ostřejší zvuk (tedy obsahuje větší podíl vyšších frekvencí v jeho spektru), který je i zase o něco ostřejší v počáteční náběhové fázi, což odpovídá zase fázím *Attack* a *Decay* ADSR obálky. A právě *Attack* a *Decay* fáze modulující mezní frekvenci dolní propusti by různě utlumovali horní část spektra signálu. Poté piáno nějakou dobu zní, kdy barva tónu je relativně konstantní, tak jako je fáze *Sustain*. A po puštění klávesy se zvuk utlumí, cože se projeví ne jenom v poklesu jeho hlasitosti, ale i v poklesu jeho „ostrosti“, což tedy aproximuje fáze *Release*. Použití ADSR obálky nebo obecně i jiných obálek, se nelimituje pouze na použití modulace amplitudy, výšky a mezní frekvence filtru. Jsou to ale asi nejčastější parametry, které se pomocí obálek modulují.

Pro další škálování celé obálky a tedy i její maximální amplitudy, se používají jiné parametry, které v podstatě modulují amplitudu obálky. Ty se na různých syntezátorech jmenují různě a ovlivňují intenzitu působení obálky na parametry, které obálka moduluje. Jedná se většinou o proměnou, která zůstává po dobu znění tónu konstantní, než nějaký výstup jiného generátoru. Dobrým příkladem je třeba rychlost úderu klávesy, která

může být označena například jako „Note ON Velocity Sensitivity“ nebo jinak podobně. Názvosloví těchto parametrů většinou není nějak zažité nebo standardizované. (V syntezátorech se často dynamika hry mění s rychlostí úderu klávesy klaviatury. Dražší klaviatury ale mají i snímače tlaku.)

5 Nízkofrekvenční oscilátory

V této části textu věnované nízkofrekvenčním oscilátorům jsem čerpal z [5] a [9]. Jak název napovídá, jedná se o oscilátory pracující při nízkých frekvencích. V kontextu se syntezátory to jsou ty frekvence, které jsou nižší než je slyšitelné spektrum lidského ucha, tedy frekvence menší než 20 Hz. V angličtině se blok syntetizéru nízkofrekvenčního oscilátoru označuje *LFO* (Low Frequency Oscillator).

Jedná se o generátory základních tvarů periodických signálů, jako je sinus, obdélník, pila nebo trojúhelník. Opět se nejedná o signál, stejně jako bylo v případě generátorů obálek, který by byl slyšet ve výstupním zvuku syntetizéru. Jde o řídící signál, který se používá k modulaci parametrů jiných částí syntetizéru. Ve srovnání s generátory obálek se většinou nespouštějí například stiskem klávesy, ale prostě jak se jednou zapnou, tak přiřazený parametr setrvale modulují. Nízkofrekvenční oscilátory se používají například k simulaci charakteru skutečných tónů, nebo vytváření zajímavých efektů.

Například modulací výšky tónu dostaneme vibráto. Což je efekt a technika používaná v mnoha hudebních nástrojích, jako jsou housle, kytara, a jiné. Nebo při velmi nízkých frekvencích a hloubky modulace („síly“ modulace) oscilátoru se bude spíš zdát, že se nástroj v čase mírné rozlad'uje. Což není až tak neobvyklá vlastnost skutečných tónů, kdy například flétnista není schopen udržet skutečně konstantní proud vzduchu do nástroje.

Modulací hlasitosti výstupního zvuku dostaneme efekt tremola. Oscilátor o nízké frekvenci a hloubce modulace ovlivňující hlasitost tónu může trochu oživit například uměle vygenerované tóny. Smysl tohoto efektu je opět vidět ve srovnání se skutečnými zvuky nástrojů, kdy například houslista zase nehraje s absolutně konstantní hlasitostí.

Další obvyklou možností je modulace mezní frekvence filtru. Popsat výsledný akustický efekt takové modulace je trochu obtížný. Tón se jakoby v chvěje, mění totiž svou barvu zvuku. Například s použitím dolní propusti se tón „rozjasňuje“ a „zastírá“, což odpovídá otevírání a zavírání filtru, a jeho různému tlumení vyšších frekvencí spektra signálu. Zase se jedná o simulaci chování skutečných tónů, v tomto případě například tón flétny se chová podobně. S použitím vysoké hloubky modulace tyto metody sice zase působí uměle, ale můžou vytvářet zajímavé efekty.

Výše popsané parametry, které lze pomocí nízkofrekvenční filtrů modulovat, patří mezi standardní použití tohoto typu generátoru. Zase se nemusí jednat o jediná použití. Jediným omezením při modifikaci nejrůznějších parametrů je samotný návrh syntezátoru. Příkladem takové trochu nezvyklé modulace může být například parametr střídy obdélníkového generátoru subtraktivní syntézy, který mění barvu tónu obdélníkového signálu. Výsledný efekt by byl trochu podobný (ale ne stejný) modulaci mezní frekvence filtru.

5.1 Softwarová implementace nízkofrekvenčních oscilátorů

Implementace generátoru sinusovky je poměrně jednoduchá a popsal jsem ji v poznámce k implementaci pro aditivní syntézu na stránce 10. Dále bude následovat popis tvorby ostatních základních tvarů vln, tedy pily, trojúhelníku a obdélníku. Je dobré poznamenat, že dále popsaný způsob generování těchto vln je generuje ve své ideální podobě. Tedy na-

příklad obdélník bude mít strmý nárůst a pokles amplitudy mezi hodnotami 1 a -1. Tyto strmé hrany z pohledu frekvenčního spektra obsahují i vysoké frekvence, které mohou přesahovat $\frac{1}{2}$ vzorkovací frekvence, a tedy mohou způsobovat aliasing signálu. Proto se tento způsob generování těchto signálů moc nehodí k použití pro tvorbu signálů, které by přímo zněly na výstupu syntetizérů. Dají se ale použít jako signály nízkofrekvenčních generátorů, jejichž signály jsou pouze řídícím signálem pro modulaci jiných parametrů.

5.1.1 Pilový signál

Tyto jednoduché vlny lze v podstatě počítat přímo. Začneme generováním pilového signálu, kde je v podstatě klíčem správně spočítat inkrement amplitudy signálu. Počet vzorků signálu dlouhého jednu vteřinu je dán vzorkovací frekvencí f_{vz}. Z frekvence pilového signálu f můžeme spočítat, že pro jednu periodu pily je třeba f_{vz}/f vzorků. Amplituda pily se bude pohybovat v intervalu $[-1, 1]$, což dělá rozdíl hodnoty amplitudy roven 2. Proto inkrement amplitudy za jeden vzorek spočítáme [5]:

$$inkrement = \frac{2}{f_{vz}/f} = \frac{2f}{f_{vz}} \tag{4}$$

Akumulátor fáze, který bude držet aktuální hodnotu amplitudy signálu, se bude hodnotou pohybovat také v intervalu $[-1, 1]$. Bude začínat s hodnotou -1 a pro každý další vzorek se bude inkrementovat hodnotou vypočtenou ze vzorce č.4. Když akumulátor přesáhne hodnotu 1, tak mu buď znova přiřadíme hodnotu -1 nebo jej snížíme o 2. To uděláme jednoduše jednou podmínkou v programu. V této části se samozřejmě objevuje malá zaokrouhlovací chyba s periodou signálu a maximální hodnotou amplitudy pily. Avšak při nízkých frekvencích pilového signálu je tato chyba zanedbatelná.

5.1.2 Trojúhelníkový signál

Trojúhelníkový signál můžeme vypočítat pomocí následujícího vztahu [5]:

$$y = 1 - \left| \frac{2(\phi - \pi)}{\pi} \right| \tag{5}$$

Kde ϕ je fáze, která se pohybuje v intervalu $[0, 2\pi]$ a výsledná amplituda bude v mezích intervalu $[-1, 1]$. Proto fázový akumulátor v programu bude také uchovávat fázi v intervalu $[0, 2\pi]$. Je třeba spočítat fázový inkrement za jeden vzorek, to lze provést podobně, jak jsme provedli pro pilový signál. Tedy při frekvenci trojúhelníkového signálu f a vzorkovací frekvence f_{vz} lze spočítat, že pro jednu periodu trojúhelníkového signálu je třeba f_{vz}/f vzorků. Rozdíl fáze v intervalu $[0, 2\pi]$ je roven 2π, a proto fázový inkrement bude roven [5]:

$$fazovy\_inkrement = \frac{2\pi}{f_{vz}/f} = \frac{2\pi f}{f_{vz}} \tag{6}$$

Akumulátor tedy bude začínat na hodnotě 0 a pro každý další vzorek bude inkrementován o hodnotu vypočtenou ze vzorce č.6. Hodnotu vzorku pro danou fázi spočítáme

dosažením hodnoty akumulátoru fáze do vzorce č.5. Když hodnota akumulátoru překročí hodnotu 2π, tak jej buď opět vynulujeme nebo jej snížíme o 2π. To lze zas provést pomocí jedné podmínky v programu. Opět zde vzniká malá zaokrouhlovací chyba pro amplitudu a periodu signálu, která je ale zanedbatelná pro nízké frekvence trojúhelníkového signálu.

5.1.3 Obdélníkový signál

Pro obdélníkový signál v podstatě stačí kontrolovat, jestli fáze překročila hodnotu, při které se má amplituda překlápět z hodnoty 1 do -1. Mějme tedy fázový akumulátor, jehož hodnota se pohybuje v intervalu $[0, 2\pi]$. Fázový inkrement se spočítá stejně, jak je uvedeno pro trojúhelníkový signál, tedy ze vzorce č.6. Akumulátor tedy inkrementujeme tímto fázovým inkrementem, a když dosáhne hodnoty 2π, tak jej vynulujeme nebo snížíme o 2π. Hodnotu amplitudy pro dané vzorky budeme v podstatě přiřazovat pomocí podmínky v programu. Chceme tedy například, aby se obdélníkový signál překlápěl v polovině své periody. To tedy znamená, že pokud je hodnota akumulátoru fáze menší nebo rovna $\frac{2\pi}{2} = \pi$, tak hodnota amplitudy vzorku je rovna 1. Pokud hodnota akumulátoru fáze je větší než π, hodnota amplitudy vzorku je rovna -1.

6 Filtr

V této obecnější části textu o filtrech jsem čerpal z [5] a [9]. Filtr je v syntezátoru založeném na subtraktivní syntéze jednou z nejdůležitějších součástí řetězce formujícího zvuk. Po tom co je vygenerován harmonicky bohatý signál (je jedno zda je to předem nahraný vzorek nebo vygenerovaná vlna), tak filtr je jeden z nejsilnějších nástrojů pro úpravu barvy vygenerovaného tónu. Filtrováním vlastně tlumíme nebo přímo ořezáváme frekvenční pásmo signálu vstupujícího do filtru. Jakou část spektra vybíráme záleží jak na typu filtru, tak na jeho mezní/mezních frekvenci/frekvencích. Respektive zdali se jedná *horní propust*, *dolní propust*, *pásmovou propust* nebo *pásmovou zádrž*, a jak jsou nastaveny jejich mezní frekvence[6].

Asi nejpoužívanějším filtrem je dolní propust. Prakticky se tak na syntezátoru objevuje parametr, který ovlivňuje mezní frekvenci filtru, v angličtině *Cutoff Frequency*. Parametr se však může nazývat trochu jinak syntezátor od syntezátoru. Předpokládejme tedy, že máme frekvenčně bohatý tón, který kromě základní frekvence[7] tónu obsahuje i vyšší harmonické frekvence[8]. Pokud se jedná o dolní propust, tak filtr tlumí frekvence vyšší než je mezní frekvence filtru. Pokud tedy tlumíme podíl vyšších frekvencí tónu, tak se tón jeví posluchači „zastřenější“. S větším podílem vyšších harmonických frekvencí se tón „rozjasňuje“, proto například jednoduché ekvalizéry audio přehrávačů mají kolečko nazvané „Brightness“. „Ostrost“ tónu je v podstatě subjektivní vnímání tónu člověkem a jelikož syntezátor vyžaduje od svého uživatele určité znalosti o tvorbě tónu, alespoň vzhledem k metodě syntézy, tak takový zjednodušující parametr „Brightness“ najdeme tak maximálně pro nastavení filtru, který je až za celým řetězcem generování a úprav tónu syntezátoru pro dodatečné korekce. Otevíráním a zavíráním filtru se většinou myslí zvětšování a zmenšování netlumeného frekvenčního pásma. Ve vztahu k dolní propusti tedy zvyšování mezní frekvence.

Ekvalizér v přehrávačích je většinou jeden filtr nebo sada filtrů, které filtrují celý přehrávaný signál. Ale v subtraktivní syntéze jde hlavně o tvarování tónu jdoucího z generátoru. Filtr je tedy většinou umístěn přímo za výstup z části s generátory. Jelikož syntezátory můžou být vícehlasé, tak logicky roste i počet generátorů a tedy i filtrů.

Filtr syntezátoru je nutné přelaďovat *relativně* k frekvenci hraného tónu. Tím docílíme podobné barvy tónu v celém rozsahu nástroje. To samozřejmě platí i pro vícehlasé syntetizéry, kde se ladí filtr přiřazený pro každý hlas[9] zvlášť.

Z frekvenčních charakteristik (amplitudy a fáze) lze vidět přesněji, jak se vlastně filtr chová.

---

[6] *Mezní frekvence* je frekvence, kdy vstupní signál je utlumen o určitou hodnotu, většinou definovanou v decibelech. Tato „určitá hodnota“ se může lišit v různé literatuře a v jakém kontextu se filtr vyskytuje, potažmo i v jakém odvětví se objevuje. Nejvíce zažitá hodnota poklesu signálu relativně k jeho úrovni na vstupu je -3 dB a s touto hodnotou je i definována mezní frekvence v tomto textu.

[7] Nejnižší frekvence tónu.

[8] Celočíselné násobky základní frekvence.

[9] Pojem hlas se používá i v syntetizérech a jeho význam je podobný. Jedná se o tón vytvořený generátory a vytvarován filtrem ještě před vstupem do bloku s efekty.

7 Digitální filtr a jeho výběr

Tématem digitálních filtrů se sice zabývají i knihy o základech DSP, ale zároveň jsou napsány i celé knihy zaobírající se jen samotným návrhem digitálních filtrů. Tato část textu se tedy jen relativně povrchně dotkne teorie jeho návrhu. Spíš jen popíše proč byl daný typ filtru vybrán.

7.1 Vlastnosti digitálních filtrů

V této části textu věnované vlastnostem digitálních filtrů jsem čerpal z [8]. Mnoho vlastností je podobných nebo stejných vlastnostem analogových filtrů. Výkonnost filtru se hodnotí ve dvou doménách, v *časové doméně* a *frekvenční doméně*. Nás budou zajímat zejména filtry s dobrými vlastnostmi ve frekvenční doméně, ale myslím, že je podstatné uvést trochu věci do obrazu.

Pokud sledujeme jak se filtr chová v *časové doméně*, sledujeme jak zkresluje vstupní signál a jeho průběh v čase na výstupu. Mějme například na vstupu obdélníkový signál. Teď záleží, jak se obdélníkový signál na výstupu změní. Důležité je jestli hrany přestřelují svou původní maximální a minimální hodnotu nebo rychlost náběhu a sestup hrany. Filtry určené k filtrování signálu v časové doméně tedy například signál vyhlazují od šumu tak, že se snaží co nejlépe zachovat tvar původního signálu bez šumu. Takové filtry se dají najít například i na digitálních osciloskopech. Filtry zaměřené na filtrování signálů v časové doméně mají většinou špatné vlastnosti ve frekvenční doméně.

V syntezátoru nemáme potřebu vyhlazovat generovaný signál, ale utlumovat nějaké jeho frekvenční pásma. Proto nás hlavně zajímají filtry s dobrými vlastnostmi ve *frekvenční doméně*. Tedy například strmost útlumu, velikost útlumu a případné zvlnění v amplitudové charakteristice, které neodpovídají ideální charakteristice filtru.

Některé typy filtrů můžou například mírně zesilovat frekvence poblíž mezní frekvence filtru. Tento jev samozřejmě není žádoucí, ale občas poskytuje filtru větší strmost útlumu. U syntezátorů toto zkreslení nemusí vždy nutně vadit. Záleží sice na typu syntezátoru, ale u většiny jde spíše o to generovat nějak zajímavý tón, většinou bohatý na harmonické složky. Některé syntezátory dokonce mají parametr, který toto zkreslení kolem mezní frekvence filtru ovlivňuje a považuje se jako součást modelování zvuku.

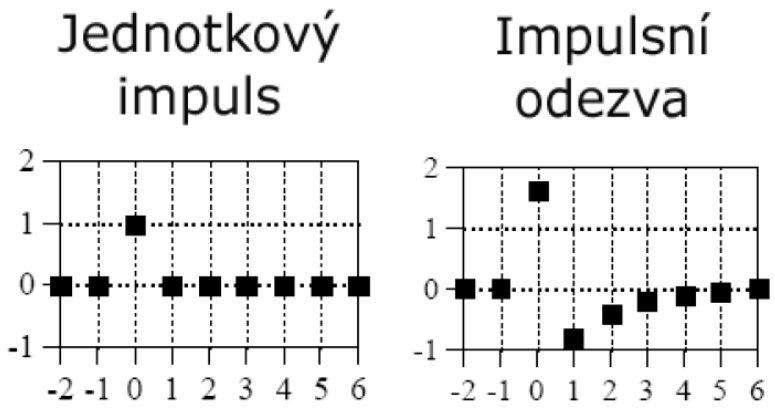
A samozřejmě bychom i měli zvážit rychlost, s jakou digitální filtr dokáže vstupní signál zpracovat. Filtr vybíráme tedy podle toho, pro jaký účel bude použit. Jelikož nelze vytvořit naprosto ideální digitální filtr, který by byl rychlý a zároveň výkonný (v časové nebo frekvenční doméně), tak vznikla spousta typů digitálních filtrů, které poskytují různý kompromis mezi dobrými a špatnými vlastnostmi.

7.2 Typy digitálních filtrů

Jedním ze základních rozdělení digitálních filtrů je podle impulzní odezvy. Existují tedy dvě hlavní kategorie filtrů, filtry s *konečnou impulzní odezvou* a filtry s *nekončenou impulzní odezvou*.

7.2.1 Filtry s konečnou impulzní odezvou

V této části textu věnované FIR filtrům jsem čerpal z [8] a [3]. Filtr s *konečnou impulzní odezvou* je angličtině označován jako *FIR* (*Finite Impulse Response*) filtr. Jak název napovídá, má konečnou impulzní odezvu. *Impulzní odezva* popisuje, jak se systém zpracovávající signál zachová pokud na vstupu bude jednotkový impuls[10] (také nazýván delta funkce). Příklad jednotkového impulsu a impulsní odezvy lze vidět na Obrázku 8 [8]. Impulzní odezva může být popsána rovnicí, která v digitální podobě popisuje hodnoty jednotlivých vzorků impulzní odezvy. Může být však i výsledkem nějakých složitějších analýz signálu a vzorkování určitých (jsou zde jistá omezení vázající se k *lineárním systémům*) skutečných analogových obvodů. Vzorkovat a analyzovat výstupy analogových filtrů by bylo ale nešikovné a značně omezující co se nastavitelných parametrů týče. V podstatě by nešlo nic změnit, tedy nešlo by měnit ani mezní frekvenci filtru, kterou potřebujeme v syntezátoru přelad'ovat. Existují však rovnice přímo popisující impulzní odezvu filtru. Většinou se jedná pouze o vzorec pro výpočet impulzní odezvy dolní propusti, protože další typy filtrů se dají z dolní propusti odvodit.



Obrázek 8: Ukázka impulzní odezvy

Rovnice vycházejí ze sinc funkce $\mathrm{sinc}(x) = \frac{\sin(x)}{x}$ [8], která reprezentuje ideální impulzní odezvu dolní propusti. Sinc není definovaná pro x=0, ale v programu se toto ošetřuje tak, aby $\mathrm{sinc}(0) = 1$, což je odvozeno z toho, že sinc se z obou stran blíží k 1. (Limita $\frac{\sin(x)}{x}$ pro x jdoucí k nule je rovna jedné.) Sinc funkce je ale definovaná v záporných i kladných hodnotách až do nekonečna. Proto se používá pro impulsní odezvu jen její část. Nemůžeme ji jen tak jednoduše useknout, musíme udržet její osovou symetrii kolem x=0 a konce zahladit nějakým oknem. Těmto filtrům se v angličtině říká *Windowed-Sinc* filtry. Okno je nějaká funkce, která k jeho koncům klesá k nule. Oknem se vybraná část výsledku sinc funkce násobí, a tím se její konce zahladí k nule na daný počet vzorků.

[10] První vzorek vstupu je o hodnotě jedna, zbytek je nula. (Pohybujeme se v digitálních systémech, ale existuje i analogová varianta.)

Oken existuje více, ale běžně používaná jsou *Blackamnovo* a *Hammingovo* okno. Mají trochu různé vlastnosti, například Hammingovo okno poskytuje strmější útlum, zatímco Blackmanovo okno má silnější útlum. Příklad výpočtu Blackmanova okna pro i-tý vzorek pro M vzorků celkem (M ještě musí být liché číslo) je vzorec č.7 [8], pro Hammingovo okno vzorec č.8 [8].

$$w[i] = 0,42 - 0,5\cos(2\pi i/M) + 0,08\cos(4\pi i/M) \tag{7}$$

$$w[i] = 0,54 - 0,46\cos(2\pi i/M) \tag{8}$$

Impulzní odezva filtru se v anglické literatuře také často nazývá *kernel* filtru a lze nalézt tuto terminologii i v některých českých publikacích. To, že filtr má konečnou impulsní odezvu prakticky znamená, že se dá taková odezva vzorkovat a použít v konvoluci s originálním signálem ve výpočetní technice. Konvoluce impulzní odezvy/kernelu se zdrojovým signálem je vlastně podstatou FIR filtrů, a proto se občas nazývají konvoluční filtry.

Konvoluce je ale velmi drahá operace, pokud se provádí klasicky v *časové doméně*[11]. Pro klasickou konvoluci vstupního signálu $x[n]$ a impulzní odezvy $h[n]$, tedy $x[n] * h[n] = y[n]$, je složitost algoritmu výpočtu rovna součinu délek signálu $x[n]$ a $h[n]$.

Zjednodušeně řečeno se dá říct, že čím delší je impulzní odezva filtru, tím lepší má výkon (strmější útlum, menší zkreslení, a tak dále). Jelikož nemůžeme moc ovlivnit délku vstupního signálu, tak pro FIR filtry platí, čím výkonnější/lepší filtr, tím větší je výpočetní složitost takového filtru.

Existuje ale algoritmus, který dokáže konvoluci urychlit. Jedná se *rychlou Fourierovu transformaci*, v angličtině nazývanou *Fast Fourier Transform* (*FFT*) a její využití v tzv. rychlé FFT konvoluci. *FFT* je efektivní algoritmus pro výpočet diskrétní Fourierovy transformace. Jedná se o algoritmus, který dokáže vstupní signál přetransformovat z reprezentace v časové doméně do frekvenční domény. Jinak řečeno navzorkovaný signál dokáže přetransformovat do podoby reprezentující frekvenční spektrum původního signálu. Což odpovídá rozkladu signálu na kosinusové a sinusové složky. Podstatné je, že násobení[12] signálů reprezentovaných ve frekvenční doméně odpovídá jejich konvoluci v doméně časové. (Oba signály musí být převedeny, tedy i impulsní odezva. Ta ale stačí samozřejmě transformovat jen jednou.) Tento princip rychlé FFT konvoluce snižuje náročnost výpočtu konvoluce. Samotný FFT algoritmus má samozřejmě taky svou výpočetní náročnost. Signál se musí navíc transformovat dvakrát, jednou do frekvenční domény a podruhé zpátky do časové domény. Proto se rychlá FFT konvoluce používá pro impulsní odezvy o délce zhruba 40-80 vzorků a více. Délka impulsní odezvy, pro kterou se rychlou FFT konvoluci už vyplatí použít, samozřejmě ještě záleží na implementaci a HW.

Nejjednodušším filtrem, který se také většinou řadí mezi FIR filtry je i filtr určený k vyhlazování signálu v časové doméně. V angličtině označován jako *Moving Average Fil-*

[11] V časové doméně je signál reprezentován svými hodnotami v závislosti na čase. Tedy co běžně zobrazí osciloskop když se díváme na obdélníkový signál.

[12] Násobení ve frekvenční doméně má svou vlastní definici, je tedy trochu jiné než klasické násobení čísel a záleží v jaké formě je výstup algoritmu Fourierovy transformace.

ter. Jak název naznačuje, jedná se v podstatě o průměrování sousedících hodnot. Čím více hodnot zprůměrujeme, tím méně je šumu, ale o to větší je zkreslení signálu. Dále nemá cenu se těmito filtry zabývat, protože nás zajímají hlavně filtry s dobrými vlastnostmi ve frekvenční doméně.

Windowed-Sinc tedy mohou nabídnou dobrou strmost útlumu, velikost útlumu a malé zkreslení na úkor větší délky impulsní odezvy. V kombinaci s rychlou FFT konvolucí dosáhneme i lepšího výkonu. Ale pokud požadujeme dostatečnou strmost a sílu útlumu, může se zdát, že tento typ filtru je pořád poněkud výpočetně náročný. Existují ale filtry s *nekonečnou impulsní odezvou*, které fungují trochu jinak, protože nelze již s impulsní odezvou konvolvovat na počítači. Jejich výhodou je, že jsou velmi rychlé, ale ostatní jejich vlastnosti jsou horší oproti *Windowed-Sinc* filtrům. *Windowed-Sinc* filtry se používají i v real-time aplikacích, pokud máme dost výkonu nebo nepotřebujeme takovou strmost a velikost útlumu (např. mp3 přehvávče). V syntezátorech je ale většinou filtrů více nebo zde probíhají další výpočetně náročné (pro real-time běh) procesy. Proto se zdají být filtry s nekonečnou impulzní odezvou vhodnější, hlavně kvůli své rychlosti. Je možné, že některé syntezátory používají Windowed-Sinc filtry (výrobci se moc často nechlubí přesnou implementací filtrů), ale pro můj syntezátor by byl výkonný Windowed-Sinc filtr plýtváním prostředků. Navíc víceméně téměř nutnost naprogramovat FFT zvětšuje obtížnost implementace.

7.2.2 Filtry s nekonečnou impulzní odezvou

V této části textu věnované IIR filtrům jsem čerpal z [8] a [3]. Jak již z názvu plyne, jedná se o filtry, jejichž impulzní odezva je nekonečně dlouhá. V angličtině jsou označovány jako *IIR* (infinite impulse response) filtry. Ale zkratka *IIR* se dá najít i v česky psané literatuře o číslicových filtrech. Nekonečnou impulzní odezvu samozřejmě nemůžeme použit v konvoluci ve výpočetní technice. Tyto typy filtrů tedy fungují jinak než *FIR* filtry. *IIR* filtry se také označují jako *rekurzivní filtry*. Nejedná se o rekurzi ve stejném smyslu, který by napadl leckterého informatika, ale používají zpětné vazby signálu. Tedy k výpočtu výstupního signálu filtru se nevyužívají pouze vzorky vstupního signálu, ale i vzorky výstupního signálu už spočtené.

Jejich chování se dá popsat *diferenční rovnicí*, přitom v DSP je zvykem tyto rovnice psát trochu jinak než pro běžné posloupnosti v matematice. Především je zvykem vstupní signál označovat $x[n]$ a výstupní signál $y[n]$. Taková rovnice může vypadat například takto [8]:

$$\begin{aligned} y[n] = a_0x[n] + a_1x[n-1] + a_2x[n-2] + a_3x[n-3] + \dots \\ +b_1y[n-1] + b_2y[n-2] + b_3y[n-3] + \dots \end{aligned} \tag{9}$$

Kde koeficienty $\{a_0, a_1, a_2, a_3, \dots\}$ a $\{b_1, b_2, b_3, \dots\}$ jsou konstanty. Rovnice v podstatě popisuje výpočet n-tého vzorku. Tak jako je ve vzorci č.9, koeficienty $\{a_0, a_1, a_2, a_3, \dots\}$ násobí odpovídající hodnoty vzorků vstupního signálu x. A koeficienty $\{b_1, b_2, b_3, \dots\}$ zase násobí odpovídající hodnoty vzorků výstupního signálu y. Koeficient b_0 chybí, protože ten by násobil hodnotu $y[n]$, tedy hodnotu aktuálně počítaného výstupního vzorku.

Dá se tedy říct, že b_0 je rovno jedné. Koeficienty „a“ a „b“ v podstatě definují chování filtru.

Rovnice rekurzivních filtrů jsou většinou v praxi dlouhé jen pár koeficientů (řekněme do desíti). Jelikož se ve výpočtu objevuje jen násobení a sčítání, a délka těchto rovnic není příliš dlouhá, tak se tyto typy filtru považují za velmi rychlé. Velmi zjednodušeně by se dalo říct, že delší rovnice s více koeficienty může lépe popsat průběh filtru, tedy i filtrů s větší strmostí a intenzitou útlumu.

Prakticky je ale počet koeficientů omezen, protože čísla s plovoucí řádovou čárkou jsou v počítači implementována s omezenou přesností. Omezená přesnost čísel způsobuje určité zaokrouhlovací chyby, které můžou snadno způsobit nestabilitu filtru. Menší zvýšení počtu použitých koeficientů lze zajistit například použitím plovoucích čísel s dvojitou přesností (místo proměnných `float` použít `double`), to ale často degraduje výkon. Přesnost čísel nám omezuje i rozsah hodnot daných koeficientů, což se prakticky projevuje tak, že mezní frekvence těchto typů filtrů může být jen v určitém použitelném rozsahu. Také obtížnost designu filtru roste s počtem jeho koeficientů.

Design filtru je tedy dán počtem a hodnotami koeficientů filtru, které ovlivňují jeho impulzní odezvu. Vztahy pro výpočet koeficientů filtru s určitou impulzní nebo *frekvenční odezvou*[13] jsou dány matematickou technikou nazývanou *Z-transformace*. *Z-transformace* nám umožňuje například konverzi *frekvenční odezvy* na koeficienty nebo slučování několika stupňů filtrů do jednoho filtru.

V praxi se totiž většinou pro filtry s vyšší strmostí a intenzitou útlumu používá kaskáda několika stupňů filtru za sebou. Tedy signál se jakoby filtruje několikrát. Navrhne se tedy jen jeden stupeň filtru (v podstatě jeden jednodušší filtr s méně koeficienty), a pak se celá kaskáda těchto stupňů může pomocí *Z-transformace* sloučit (do jednoho filtru s více koeficienty). Používá se ale i obyčejné několikanásobné filtrování, kdy opravdu jen několikrát přefiltrujeme vstupní signál. Několikanásobné filtrování nelze dělat do nekonečna, zase jsme omezeni různými faktory, jako je přesnost čísel s plovoucí čárkou. Záleží na konkrétním designu filtru, ale v praxi se provádí několikanásobné filtrování přibližně maximálně třikrát, než filtr začne být nestabilní.

Je tedy jasné, že návrh těchto filtrů je poměrně složitý. A návrh rekurzivního filtru od začátku do konce je v podstatě mimo rozsah této práce. Pokud se tedy nezabýváte přímo pouze návrhem filtrů, tak existují už hotová řešení, která se víceméně už jen opisují. Například existují tabulky koeficientů, z kterých lze převzít hodnoty pro určité hodnoty mezní frekvence. Což není často ideální, protože hodnoty tabulek definují hodnoty koeficientů jen pro určité mezní frekvence. Další možností je použít nějakou proceduru/funkci z nějaké knížky DSP, která vám koeficienty pro určitou mezní frekvenci a počet stupňů spočítá.

[13] *Frekvenční odezva* systému nám udává, jak se změní frekvenční spektrum vstupního signálu na výstupu po průchodu systémem. Dá se vyjádřit například frekvenční charakteristikou. *Frekvenční odezva* se dá konvertovat na *impulzní odezvu* a naopak.

7.2.2.1 Jednoduché rekurzivní filtry

Existují základní rekurzivní filtry, které odpovídají svými vlastnostmi jednoduchým jednostupňovým RC filtrům, které známe ze základů elektroniky například ze střední školy. Koeficienty pro dolní propust se dají spočítat ze vztahu [8]:

$$\begin{aligned} a_0 &= 1 - x \\ b_1 &= x \end{aligned} \tag{10}$$

A koeficienty pro horní propust se dají spočítat ze vtahu [8]:

$$\begin{aligned} a_0 &= (1 + x)/2 \\ a_1 &= -(1 + x)/2 \\ b_1 &= x \end{aligned} \tag{11}$$

Parametr x je hodnota v intervalu $[0, 1)$, která v podstatě udává průběh filtru. Například ve vztahu č.10 jde dobře vidět, že x udává jak moc předchozí vypočtený vzorek výstupu filtru ovlivňují vzorek aktuální. Diferenční rovnice pro dolní propust by tedy měla tvar $y[n] = a_0x[n] + b_1y[n - 1]$. Pokud by například x bylo rovno 0, po dosazení bychom dostali $y[n] = x[n]$, tedy neprobíhala by žádná filtrace, jen by se signál vstupu kopíroval na výstup. Nás ale zajímá hodnota x pro danou mezní frekvenci f_m, to se dá spočítat ze vztahu [8]:

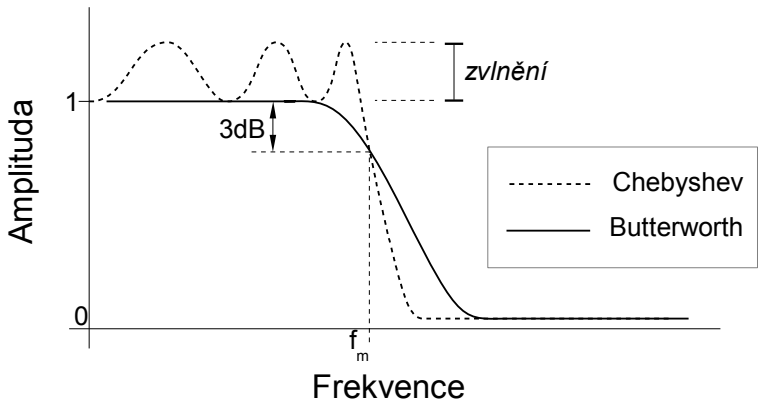
$$x = e^{-2\pi f_m} \tag{12}$$

7.2.2.2 Chebyshevův a Butterworthův filtr

V této části textu věnované Chebyshevově a Butterworthově filtru jsem čerpal z [8]. Chebyshevův filtr využívá zvlnění části své frekvenční charakteristiky k dosažení strmějšího útlumu signálu. Toto zvlnění je vidět na ilustračním obrázku č.9, kde je frekvenční charakteristika Chebyshevova filtru nakreslena čárkovaně a f_m je mezní frekvence. Zvlnění Chebyshevova filtru je parametr zadaný v procentech. Dalším nutným parametrem je samozřejmě mezní frekvence filtru. (Chebyshev je ruské jméno, v české literatuře se objevuje i překlad „Čebyševův filtr“.)

Existují dva typy Chebyshevova filtru, jeden má toto zvlnění přítomno v části frekvenční charakteristiky, která signál propouští. A druhý typ zase obsahuje toto zvlnění v části, která signál tlumí. Na obrázku č.9 je tedy frekvenční charakteristika prvního typu. Pokud parametr zvlnění nastavíme na nulu, vzniká Butterworthův filtr, jehož frekvenční charakteristika je na obrázku vyznačena plnou čarou. Je tedy jasné, že stačí navrhnout Chebyshevův filtr, který v případě potřeby můžeme změnit na Butterworthův. S rostoucím zvlněním roste také strmost útlumu filtru (také lze vidět na obrázku). Přijatelným zvlněním pro většinu aplikací je 0,5 % [8], můžeme však na úkor kvality filtru zvyšovat zvlnění (a tak i strmost útlumu) podle potřeby.

Návrh takovéhoto filtru je zase velmi komplikovaný, a tak většinou programátoři přepisují už hotové řešení, které spočítá koeficienty diferenční rovnice pro danou mezní frekvenci, procentuální zvlnění a typ (dolní propust, horní propust, atd.) filtru. Já jsem pro funkci mého programu, která počítá koeficienty, vycházel z pseudokódu napsaného



Obrázek 9: Ilustrační příklad frekvenční charakteristiky Butterworthova a Chebyshevova filtru

v jazyce BASIC v knize The Scientist and Engineer's Guide to Digital Signal Processing od autora se jménem Steven Smith (tedy kniha [8]). Tyto pseudokódy lze nalézt v příloze C na straně 50.

Tyto dva typy filtru jsou docela vhodné pro použití v syntezátorech. Jsou rychlé a mají lepší strmost a velikost útlumu než základní rekurzivní filtry popsané v předešlé kapitole. V některých typech syntézy (hlavně ty co generují tóny umělé) si můžeme dovolit u Chebyshevova filtru i vyšší hodnoty zvlnění. Ale měli bychom najít nastavováním jeho parametrů rozumný kompromis kvality zvuku.

7.3 Výpočetní výkon a problémy

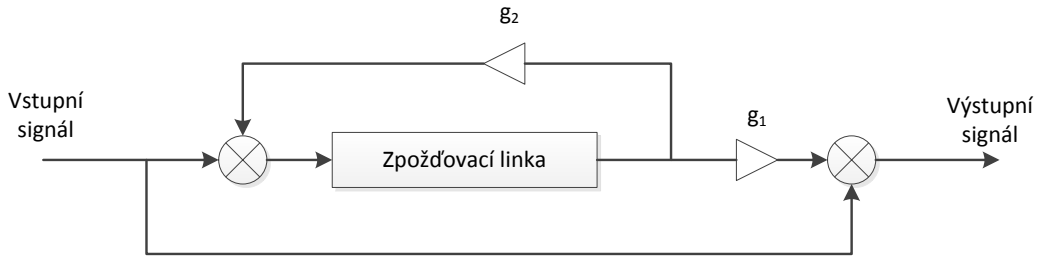
Různé techniky, jak vlastně filtrovat digitální signál, mají různé vlastnosti. Digitální syntezátor musí ale pracovat v reálném čase, a proto jedním ze základních požadavků je rychlost zpracování vstupního signálu. Návrh filtru je vždy o kompromisu mezi výkonem a rychlostí, a samozřejmě i jiných požadavků, které klademe na filtr. Přestože doba poměrně pokročila a výkony dnešních procesorů jsou obrovské v porovnání s minulostí, tak pořád není vhodné plýtvat výkonem například použitím Window-Sinc filtru v aplikaci, která má běžet v reálném čase. Window-Sinc filtr s dlouhým kernelem (impulzní odezvou) i s použitím FFT konvoluce v aplikaci, která tak velké výkony filtru nevyžaduje a potřebuje spíš rychlý filtr, je plýtváním výpočetních prostředků.

Dalším trochu jiným problémem je například fakt, že tento softwarový syntezátor nepoběží na operačním systému reálného času. Což znamená hlavně to, že odezva systému a intervaly, v kterých se zpracování dat provádí, není odhadnutelná. Toto lze pozorovat i u komerčního softwaru, kdy může docházet k „trhání" (hluchá místa) nebo „zpomalování" generovaného nebo přehrávaného zvuku a může pomoct třeba nastavování vyšších priorit procesu programu.

8 Jednoduché echo a delay efekt

V této části textu věnované efektu echo a delay jsem čerpal z [5][7], hlavně pro jejich definice a sestavení schématu, odvozování amplitud a jejich korekce je vlastní. Mnoho syntezátorů obsahuje ještě blok s efekty. Efekty se aplikují až nakonec, tedy až zvuk projde všemi fázemi generování a úprav zvuku řetězce syntezátoru. Proto parametry efektů už většinou nelze modulovat žádnými generátory řídících signálů syntezátoru nebo rychlostí úderu klávesy. Tento blok tedy působí většinou samostatně.

Jedny ze základních efektů jsou efekty *echo* a *delay*. Bohužel názvy těchto efektů se často v terminologii různých knih a výrobců zaměňují a mísí dohromady. Tedy možnou definicí echa je, že se jedná o prosté opakování signálu s určitým zpožděním na výstupu. Zatímco efekt delay se může chovat stejně jako echo, tedy pouze signál jednou zopakovat s určitým zpožděním, nebo se signál může ozývat s určitým zpožděním několikrát až do ztracena.



Obrázek 10: Schématické znázornění cesty signálu delay efektu

Na obrázku č.10 lze vidět schématické znázornění delay efektu. Značky zesilovačů g_1 a g_2 znázorňují útlum signálu. Pro začátek si představme, že g_2 je nastaveno na hodnotu nula, tedy zpětná vazba nepropouští žádný signál. Zatím tedy budeme tuto zpětnou vazbu ignorovat. Vstupní signál se v podstatě rozděluje do dvou větví. Jedna vstupuje do *zpožďovací linky*, která vytváří zpoždění signálu. Tento zpožděný signál se ještě škáluje hodnotou g_1 a putuje do sumačního členu, který jednoduše signál sečte s nezpožděným signálem z druhé větve. Výstupem tedy bude jednoduché *echo*, kdy se signál zpožděný a utlumený hodnotou g_1 ozve právě jednou. Zpoždění signálu je samozřejmě dáno *zpožďovací linkou*, která je většinou implementována s nastavitelným zpožděním.

Mějme vstupní signál, jehož amplituda se pohybuje v intervalu $[-\frac{1}{2}, \frac{1}{2}]$. Teď hodnotu g_2 nastavíme v intervalu $(0, \frac{1}{2})$ (proč tyto určité hodnoty vysvětlím dále). Tedy už zpožděný signál ze zpožďovací linky bude přiváděn s útlumem g_2 zpátky do zpožďovací linky. Útlum g_2 způsobuje, že amplituda signálu (výstup zpožďovací smyčky) přiváděného zpětnou vazbou bude menší než amplituda již uloženého signálu ve zpožďovací smyčce. To prakticky znamená, že se vstupní signál bude opakovat několikrát s klesající amplitudou až do ztracena, což odpovídá efektu *delay*. Teoreticky by se opakující signál takto ozýval do nekonečna, jen jeho hlasitost by klesla pod slyšitelnou hodnotu. Ve výpo-

četní technice jsou čísla s plovoucí čárkou implementována s omezenou přesností, a tak po určité době amplituda stejně klesne k nule.

Se špatnou volbou g_2 vzhledem k maximální amplitudě signálu na vstupu se může stát, že kladná zpětná vazba způsobí růst amplitudy signálu až do nekonečna, místo toho aby se jednotlivé ozvěny signálu ztišovali. Toto je jasné pro g_2 větší než jedna. Stává se to ale i pro hodnoty g_2 menší než jedna, protože záleží i na amplitudě vstupním signálu, který se přičítá. (Pořád mluvíme o delay efektu na obrázku č.10.) Lépe se to popisuje na příkladu.

Například mějme sinusový signál o frekvenci f, jehož amplituda se pohybuje v intervalu $[-1, 1]$. Hodnotu g_2 zvolíme rovnu $0,4$. Tón necháme znít dlouho, několikanásobně déle, než je nastavené zpoždění zpožďovací linky. Nejdříve by se tedy zpožďovací linka naplnila (jejíž délka je dána nastaveným zpožděním) signálem sinusovky ze vstupu, tedy sinusovkou s maximální amplitudou 1 a minimální amplitudou -1. Po přeškálování hodnotou g_2 by zpětnou vazbou putovala sinusovka s amplitudou v intervalu $[-0,4; 0,4]$. Může se navíc stát, že zpoždění zpožďovací linky bude zrovna nastaveno tak, že signál putující zpětnou vazbou bude mít stejnou fázi jako stále přicházející vstupní signál. Potom po sečtení zpětnovazebního signálu se vstupním signálem (tón stále zní) bychom dostali sinusovku s amplitudou v intervalu $[-1,4; 1,4]$. Amplituda signálu se tedy zvýšila. Tento signál by zase naplnil zpožďovací linku, pak znova přeškáloval přes g_2 a znova sečetl se vstupním signálem, a tak stále dokola zvyšoval svou amplitudu. Chování maximální amplitudy signálu obsaženého ve zpožďovací smyčce by se dalo tedy vyjádřit posloupností $a_n = A_{vst} + a_{n-1}g_2$, $a_0 = A_{vst}$, kde A_{vst} je maximální amplituda vstupního signálu a n odpovídá počtu naplnění/přepsání zpožďovací smyčky. (Kolikrát se naplní závisí na hodnotě zpoždění a délce signálu, ale jak hodnota maximální amplitudy ve zpožďovací smyčce přesáhne určitou mez, tak už se dokáže přesytit přes zpětnou vazbu i bez vstupního signálu.)

Proto je vhodné například udržovat amplitudu vstupního signálu v intervalu $[-\frac{1}{2}, \frac{1}{2}]$ a parametr g_2 v intervalu $(0, \frac{1}{2})$. Zase budeme vycházet z příkladu se sinusovkou v předchozím odstavci. Maximální amplituda signálu obsažená ve zpožďovací lince pro $g_s = \frac{1}{2}$ a $A_{vst} = \frac{1}{2}$ potom bude dána jako posloupnost $a_n = \frac{1}{2} + \frac{1}{2}a_{n-1}$, $a_0 = \frac{1}{2}$. Tento rekurentní vztah se dá přepsat do tvaru $a_n = 1 - 2^{-n-1} = 1 - \frac{1}{2^{n+1}}$, kde už lze vidět, že maximální amplituda signálu ve zpožďovací smyčce bude exponenciálně stoupat (při stálém znění tónu na vstupu) a blížit se hodnotě 1. Nedojde tedy k „přesycení“ zpětnou vazbou a po přestání znění tónu signál vždy postupně klesne k nule. Hodnota g_2 se volí ještě menší než $\frac{1}{2}$ kvůli zaokrouhlovacím chybám, které při výpočtu mohou vzniknout. A čím větší je hodnota g_2, tím déle se bude vstupní signál ozývat, s periodou nastavenou zpožděním.

8.1 Implementace zpožďovací linky

V této části textu věnované implementaci zpožďovací linky jsem čerpal z [5]. Pro efekty jednoduchého echa a delay potřebujeme vytvářet zpoždění signálu. Potřebujeme tedy naprogramovat *zpožďovací linku*, která vzorky signálu zpozdí. Implementace zpožďovací linky se dá provést pomocí kruhového bufferu, jehož délka bude odpovídat délce maximálního možného zpoždění, které chceme generovat. Délku bufferu tedy můžeme

spočítat jednoduše ze vzorkovací frekvence f_{vz} a času zpoždění v sekundách. Vzorkovací frekvence nám v podstatě říká, kolik vzorků tvoří jednu vteřinu signálu. Délku bufferu tedy spočteme jednoduše $f_{vz} \cdot t$.

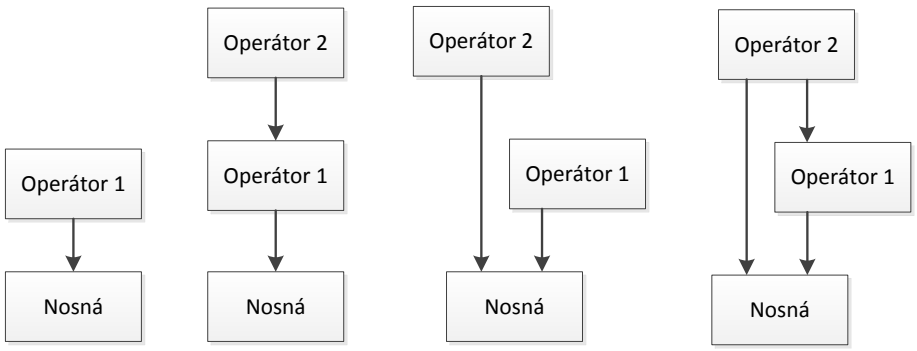
Kruhový buffer tedy bude mít nějaký ukazatel, který bude ukazovat na poslední přidaný prvek, a který se bude inkrementovat s každým přidáním vzorku. Délka bufferu představuje maximální možné zpoždění, ale my většinou potřebujeme měnit zpoždění v nějakém rozsahu. Tedy pro zpožďovací linku si budeme ještě držet proměnou, která bude reprezentovat nastavené zpoždění. Tato proměnná může být jen celočíselná, a reprezentovat zpoždění přímo v počtu vzorků (zase snadno přepočteme s $f_{vz} \cdot t$). Tento počet vzorků zpoždění nám tedy řekne, o kolik vzorků dozadu od ukazatele kruhového bufferu se nachází výsledek.

Prakticky je kruhový buffer jen obyčejné jednorozměrné pole, do kterého lze přidávat prvky donekonečna, protože kruhový buffer cyklí dokolečka a přemazává staré prvky v poli. Stejně tak si musíme dávat pozor při čtení tohoto bufferu, abychom se nesnažili dostat mimo dimenze pole (jednoduše podchytíme podmínkami). V případě potřeby zadávání zpoždění časem se může stát, že přepočet času na počet vzorků vyjde jako necelé číslo. Zde si vystačíme například se zaokrouhlováním, čímž se vyhneme zkreslení signálu nějakým typem interpolace.

9 Poznámky k implementovanému FM syntezátoru

V rámci této práce jsem naprogramoval jednohlasý FM syntezátor. Obrázek uživatelského rozhraní naleznete v přílohách na obrázku č.12 na straně 45. Názvy jednotlivých parametrů syntezátoru jsou v angličtině, protože lidé obeznámeni s nějakým syntezátorem lépe poznají, o jaký parametr se jedná z angličtiny, než z českých překladů těchto parametrů. Anglické názvy jsou tedy více zažité.

Tento FM syntezátor implementuje FM syntézu se třemi operátory, které lze kombinovat pomocí 4 různých algoritmů. Jelikož ikony uživatelského rozhraní těchto 4 algoritmů jsou poměrně malé, tak větší obrázek implementovaných algoritmů naleznete na obrázku č.11.



Obrázek 11: Algoritmy FM syntézy implementované FM syntezátorem

Dále program FM syntezátoru implementuje *ADSR obálky* k modulaci amplitudy signálu vystupujícího z generátorů FM syntézy a k modulaci mezní frekvence filtru. Samozřejmě je tedy i naprogramován filtr. Jedná se o implementaci Chebyshevova filtru (*IIR rekurzivní filtr*). Dále jsou implementovány nízkofrekvenční oscilátory, které jsou také použity k modulaci amplitudy nebo frekvence signálu. Průběh nízkofrekvenčního oscilátoru lze vybrat ze základních tvarů vln, což jsou tedy sinusovka, trojúhelník, pila a obdélník. Syntezátor také obsahuje implementaci jednoduchého *delay* efektu.

Popis jednotlivých parametrů v uživatelském rozhraní lze nalézt v uživatelské příručce, která je v přílohách na straně 46.

9.1 Použité nástroje

Projekt je vytvořen a zkompilován ve vývojovém prostředí *Microsoft Visual Studio 2010* a je programován v jazyce C++. Pro obsluhu zvukové karty používám open source knihovnu *The Synthesis ToolKit* ve verzi 4.4.4 (`http://ccrma.stanford.edu/software/stk/`) a její třídu *RtAudio*. Zbytek kódu, který vytváří výsledný zvuk syntezátoru (FM syntéza, ADSR obálky, filtr, atd.), je naprogramován vlastními silami a znalostmi popsanými v této práci.

Pro uživatelské rozhraní bylo použito open source knihovny *Qt* (`http://qt-project.org/`) ve verzi 5.0.2 pro x86 architekturu. Tato knihovna je tedy i nutná ke kompilaci projektu. Pokud je aktuální verze novější, starší verze se dají sehnat z download archivu oficiálních stránek, například tedy `http://download.qt-project.org/archive/qt/5.0/5.0.2/`. V uživatelském rozhraní se nachází i klaviatura, která je open source pluginem pro *Qt*. Název pluginu je *Virtual Piano Keyboard* a autorem je Pedro Lopez-Cabanillas, plugin byl stažen z `http://qt-apps.org/content/show.php?content=85050`. (Funkčnost všech url ověřena k datu 19. dubna 2014.)

9.2 Minimální požadavky na běh programu

Zkompilovaný program by měl fungovat na operačních systémech *Microsoft Windows Vista* a novější. (Nejnovější verze Windows v době vypracování projektu je Windows 8.1.) Jelikož byl program kompilován v *Microsoft Visual Studio 2010*, tak je zřejmě také potřeba mít nainstalovaný *Microsoft Visual C++ 2010 Redistributable Package (x86)*, pokud jej už nainstalovaný nemáte. Teoreticky by nemusel být tento balík nutný vzhledem k *.dll* knihovnám, které jsou s programem ve stejné složce, ale neměl jsem možnost program vyzkoušet na počítači bez tohoto balíku.

Co se hardwarových požadavků týče, program byl úspěšně testován i na starším notebooku s procesorem *Intel Core 2 Duo* se dvěma jádry pracujícími na frekvenci 2,0 GHz.

10 Závěr

Hlavní myšlenkou této práce bylo popsat některé základní způsoby generování zvuku a možnosti úprav jeho charakteru (barva, hlasitost, atd.), které se používají v digitálních syntetizérech a tyto znalosti uplatnit pro tvorbu softwarového syntezátoru. Implementovaný syntezátor obsahuje generátor FM syntézy, jehož operátory lze propojit pomocí čtyř různých volitelných algoritmů. Také implementuje filtr (dolní propust), generátory ADSR obálky modulující amplitudu generovaného signálu a mezní frekvenci filtru, nízkofrekvenční oscilátory s různým průběhem vlny a cíli modulace a také obsahuje jednoduchý delay efekt. Podobné možnosti se dají najít i v jiných digitálních FM syntezátorech, které se liší například výběrem modulovaných parametrů, typy filtrů nebo komplexností obálek. Mým největším přínosem je tedy samotný program FM syntezátoru, jehož návrh (typ filtru, modulované parametry, atd.) a implementace byli podřízeny tvorbě zajímavých tónů. Zároveň by měly být nastavitelné parametry implementovaného syntezátoru povědomé lidem, kteří už mají zkušenost s nějakým jiným digitálním syntezátorem.

Na implementovaný FM syntezátor lze hrát klikáním na klaviaturu v uživatelském rozhraní. Což prakticky není nejideálnější způsob hry na hudební nástroj. Proto v případě dalšího vývoje projektu by například šlo naprogramovat podporu kláves připojitelných k počítači. Nebo naprogramovat softwarový sekvencer, který by povely pro stisk a pouštění kláves generoval, a který by tak umožnil skládat hudbu generovanou syntezátorem. Naprogramovaný nástroj je jednohlasý, šlo by tedy i naprogramovat podporu pro polyfonii.

Digitální syntezátory jsou tématicky blízké digitálnímu zpracování signálů. Jedná se tedy o poměrně zajímavé praktické použití digitálního zpracování signálů pro vytváření hudebně zajímavých zvuků. Studium problematiky digitálních syntetizérů mě tedy obeznámilo s určitými základy DSP, které se v digitálních syntezátorech aplikují. Dozvěděl jsem se mnohé i o samotných hudebních syntezátorech jako takových. A doufám, že se mi tyto znalosti podařilo přenést do textu této práce a využít je v implementovaném programu.

11 Reference

[1] AIKIN, Jim. *Power tools for synthesizer programming: the ultimate reference for sound design.* San Francisco: Backbeat Books, c2004, 199 s. ISBN 08-793-0773-0.

[2] CHOWNING, John. The Synthesis of Complex Audio Spectra by Means of Frequency Modulation. *Journal of the Audio Engineering Society.* 1973, Volume 21, Issue 7. Dostupné z: https://ccrma.stanford.edu/sites/default/files/user/jc/fmsynthesispaperfinal_1.pdf

[3] KUO, Sen M., Bob H. LEE a Wenshun TIAN. *Real-time digital signal processing: implementations and applications.* 2nd ed. Chichester: John Wiley & Sons Ltd, 2006, 646 s. ISBN 04-700-1495-4.

[4] MIRANDA, Eduardo Reck. *Computer sound design: synthesis techniques and programming.* 2nd ed. Woburn, Massachusetts: Focal Press, 2002. ISBN 02-405-1693-1.

[5] MITCHELL, Daniel R. *BasicSynth: creating a music synthesizer in software.* 1st edition. Raleigh, North Carolina: Lulu.com, 2008. ISBN 978-055-7022-120.

[6] ROADS, Curtis. *The computer music tutorial.* Cambridge, Massachusetts: MIT Press, c1996. ISBN 02-626-8082-3.

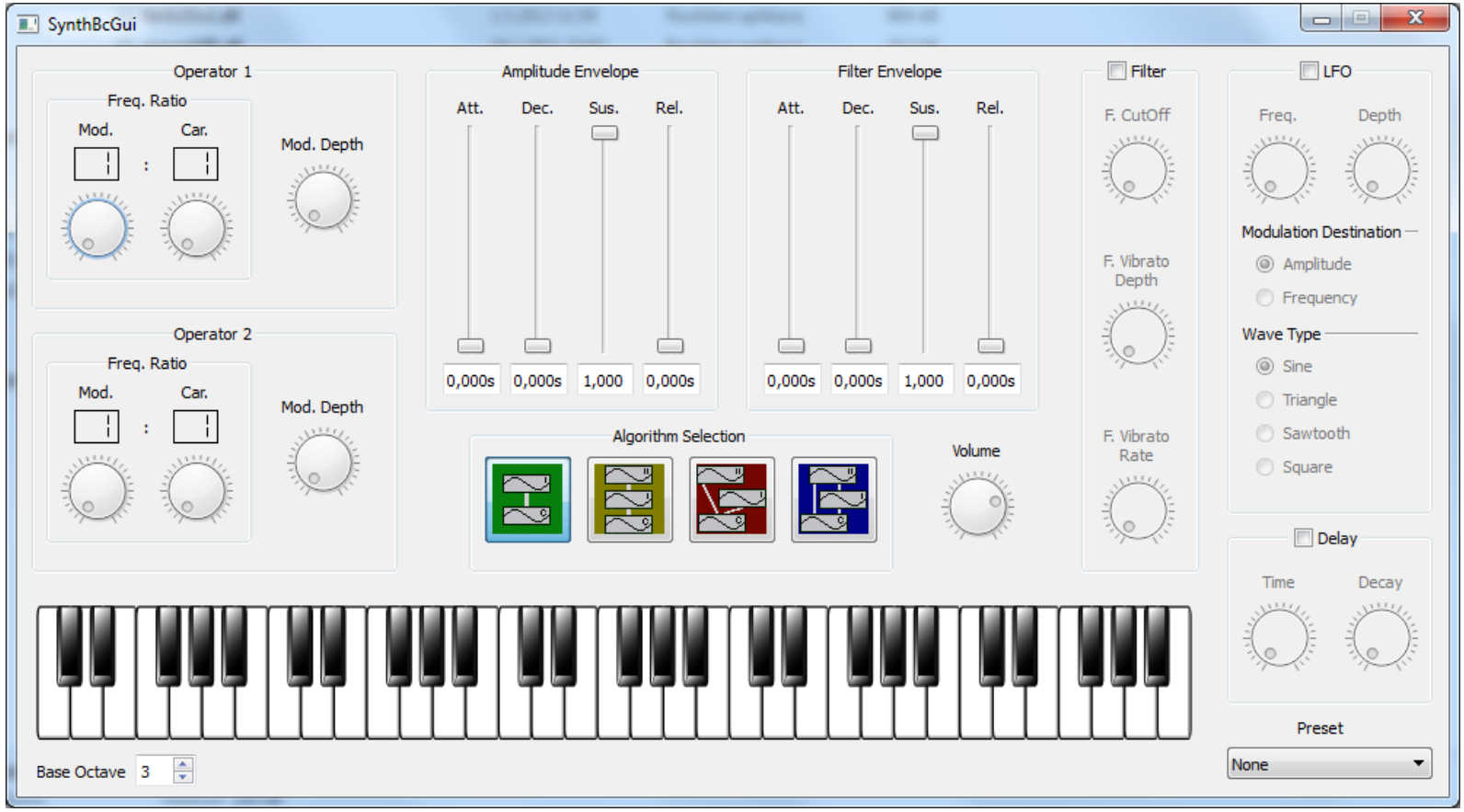
[7] SMITH, Julius O. *Physical Audio Signal Processing.* Lexington: W3K Publishing, 2010. ISBN 978-0-9745607-2-4.

[8] SMITH, Steven. *The Scientist and Engineer's Guide to Digital Signal Processing.* Second edition. San Diego, California: California Technical Publishing, 1999. ISBN 09-660-1767-6. Dostupné také z: http://www.dspguide.com/

[9] TEOCHARISOVÁ, Vanda. *SOUND DESIGN: Zvuková syntéza a tvůrčí programování zvuků v praxi.* Praha: Muzikus, 2009. ISBN 80-86253-53-4.

[10] WEISSTEIN, Eric W. Fourier Series. In: *MathWorld: A Wolfram Web Resource* [online]. c1999-2014 [cit. 2014-04-22]. Dostupné z: http://mathworld.wolfram.com/FourierSeries.html

A Další obrázky



Obrázek 12: Uživatelské rozhraní implementovaného syntezátoru

B Uživatelská příručka

Program se spouští souborem `SynthBcGui.exe`, který se nachází ve složce programu. A vypíná se jednoduše pomocí křížku pro zavírání okna. Program také obsahuje přednastavené presety, které demonstrují možnosti syntezátoru. Tyto presety ukazují nastavení určitých částí syntezátoru tak, aby šlo jejich vliv dobře slyšet na výstupu, a tak i lépe pochopit jejich funkci. Je zde i pár presetů s různými c/m poměry operátorů pro různé algoritmy FM syntézy.

Potenciometry uživatelského rozhraní lze ovládat jednoduše tahem myši. Kliknutím na potenciometr a potáhnutím myši od něj lze dostat větší poloměr pro manipulaci s potenciometrem, a tak získat vyšší citlivost pro manipulaci s hodnotami potenciometru. Potenciometry lze také ovládat kolečkem myši. Po kliknutí na potenciometr, a tak získáním jeho fokusu, lze také zvětšovat nebo zmenšovat hodnotu potenciometru šipkami nahoru a dolů, případně pomocí PageUp a PageDown přenastavovat většími skoky hodnot. Nastavení obálek lze provádět šoupátky, nebo přímo vpisováním určitých hodnot, které jsou v určitém povoleném rozsahu (max. a min. šoupátka). Šoupátka i číselníky lze také ovládat pomocí kolečka myši nebo kurzorových šipek nahoru a dolů.

Pomocí LFO lze modulovat amplitudu nebo frekvenci, a tak vytvářet efekt tremola nebo vibráta s různým průběhem podle zvoleného tvaru vlny. Delay efekt dokáže simulovat jednoduchou ozvěnu. Filtr je dolní propust a může signál filtrovat. Zároveň lze sinusovým nízkofrekvenčním oscilátorem (F. Vibrato Rate a F. Vibrato Depth) modulovat mezní frekvenci tohoto filtru, tedy měnit barvu tónu během jeho znění. Na nástroj se dá hrát klikáním na klaviaturu. Klaviatura ale podporuje i hru na klávesnici, jen je nastavená pro anglické rozložení klávesnice.

Tvorba zvuku FM syntézou je poměrně značně neintuitivní a hodně závislá na pokusech. V případě prvního algoritmu, kdy je nosná modulována jedním operátorem, je experimentování poměrně snadné. Některé „hezky“ znějící c/m poměry jsou například 1/1, 1/2, 1/3, 2/3, 3/2, 7/3. Použitelných poměrů je více, ale v praxi se tyto poměry většinou právě zjišťují zkoušením, než nějakou složitou analýzou výstupního spektra. V podstatě by měla frekvence modulátoru ladit s frekvencí nosné, podle čeho tedy nastavujeme c/m poměr. Například c/m 1/2 říká, že frekvence modulátoru bude o oktávu výše oproti nosné. Dobře tedy většinou zní hudebně ladící poměry. Kromě 1/2 třeba i tzv. čistá kvinta pro c/m rovno 2/3 a jiné hudebně ladící poměry. Přidáváním hloubky modulace (Mod. Depth) se výstupní signál subjektivně „zaostřuje“, přibývá totiž více vyšších harmonický (objevují se ale i harmonické nižší než je frekvence nosné, tedy hraného tónu).

Tento c/m poměr v podstatě i omezuje použitelný rozsah nástroje, tedy použitelnou část klaviatury. Buď se spokojíme s menším rozsahem klaviatury a nebo se tento c/m poměr musí rozumě nastavit. Třeba přibližně tak, aby frekvence modulace nebyla vyšší než řekněme desetinásobek nosné (tedy aby c/m nebylo menší než 1/10), nebo radši ještě menší. V podstatě nelze modulovat větší frekvencí, než je polovina vzorkovací frekvence a nástroj pracuje na vzorkovací frekvenci 44100 Hz. Rozsahy c/m poměrů jsou povolené záměrně velké, aby poskytovali větší množství nastavitelných poměrů (například 14/11). Změnou c/m často dochází i k změně výšky tónu a tedy i jeho vyšších harmo-

nických, a tak by mohli generované frekvence překročit polovinu vzorkovací frekvence. Proto lze klaviaturu ještě dodatečně transponovat pomocí hodnoty Base Octave. Vyšší frekvence spektra (ale i nižší) samozřejmě ovlivňuje i hloubka modulace. Pro první a třetí algoritmus FM syntézy většinou hloubka modulace nedělá problémy. Pro druhý a čtvrtý algoritmus se už občas vyplatí hloubku modulace snižovat (hlavně pro operátor 2).

Druhý algoritmus v podstatě moduluje signál dvakrát, operátor 2 moduluje operátor 1, a ten pak moduluje nosnou. V případě 2. algoritmu se vyplatí začínat nastavováním operátoru 1, kdy si nastavím pěkně znějící poměr a hloubku modulace (hloubku modulace operátoru 2 zatím nechám na nule, tím se chová tento algoritmus stejně jako 1. algoritmus), a pak teprve začnu nastavovat c/m poměr operátoru 2 a jeho hloubku modulace. Frekvence operátorů a nosné by měli mezi sebou ladit. Poměry frekvencí operátorů 1 a 2 jsou určené vzhledem k frekvenci nosné (slyšitelné) vlny. Proto například já nastavuji hodnotu poměrů frekvencí operátorů (Freq. Ratio) pro nosnou (potenciometr Car.) na stejnou hodnotu pro oba operátory, a pak zkouším měnit hodnotu c/m poměru pro modulační vlnu (potenciometr Mod.). Většinou už nehýbu s nastavením operátoru 1, který jsem už nastavil, ale pouze s nastavením operátoru 2. Dobře také fungují poměry, které odpovídají celočíselným násobkům nosné, tedy c/m poměry 1/2, 1/3, 1/4, atd. Měli by fungovat i jiné hudebně ladící poměry. Ale zase najít poměry, které zní „hezky“ je otázkou pokusů. Trochu více zde záleží i na nastavení modulační hloubky (Mod. Depth) pro operátor 2. Je dobré začínat s nižšími hodnotami, například přibližně mezi 1/4 a 1/2 potenciometru. Občas při některých c/m poměrech a vysoké modulační hloubce může být výstup podobný šumu, kdy tohle jde právě trochu napravit snížením modulační hloubky.

Pro třetí algoritmus doporučuji nastavovat operátory také postupně. Jen je jedno, který operátor nastavujeme jako první. Prakticky tyto dva operátory by měli ladit mezi sebou (jejich výstupy se sčítají) a nosnou, zde tedy zase třeba pomůže nastavit v jejich poměrech hodnotu Car. na stejnou hodnotu, a pak zkoušet měnit Mod. hodnotu. A fungují tedy i poměry odpovídající celočíselným násobkům nosné, tedy c/m poměry 1/2, 1/3, 1/4, atd. Fungují samozřejmě i jiné poměry, které mezi sebou ladí (dle hudební teorie). Pro čtvrtý algoritmus doporučuji postupovat podobně jako pro druhý algoritmus. Zbytek ovládání syntezátoru by měl být jasný z popisu jednotlivých parametrů.

B.1 Popis parametrů syntezátoru

Následuje stručný popis nastavitelných parametrů v uživatelském rozhraní naprogramovaného syntezátoru. Obrázek uživatelského rozhraní naleznete v přílohách na obrázku č.12 na straně 45.

Operator 1 Operátor č. 1 FM syntézy označovaný v algoritmech na ikonách.

- **Freq. Ratio** Zkratka pro Frequency Ratio. Pomocí parametrů *Mod.* a *Car.* lze určit poměr frekvencí mezi operátorem č.1 a nosnou.
- **Mod. Depth** Zkratka pro Modulation Depth. Udává hloubku FM modulace pro operátor 1. Jedná se v podstatě o nastavení modulačního indexu, kdy maximální hodnota odpovídá modulačnímu indexu rovnu 2.

Operator 2 Operátor č. 2 FM syntézy označovaný v algoritmech na ikonách.

Freq. Ratio, Mod. Depth Stejně jako pro operátor č. 1, jenom se váže k operátoru č. 2

Algorithm Selection Výběr algoritmu FM syntézy. Pro výběr stačí kliknout na příslušnou ikonu.

Amplitude Envelope Sekce nastavení amplitudové ADSR obálky. Ovlivňuje amplitudu vygenerovaného signálu FM syntézou s daným algoritmem.

Att. Attack fáze ADSR obálky, nastavitelné v sekundách.

Dec. Decay fáze ADSR obálky, nastavitelná v sekundách.

Sus. Sustain fáze ADSR obálky. Číslo udává úroveň amplitudy v intervalu $[0, 1]$.

Rel. Release fáze ADSR obálky, nastavitelná v sekundách.

Filter Envelope Sekce nastavení ADSR obálky filtru. Ovlivňuje mezní frekvenci filtru.

Att., Dec., Sus., Rel. Stejně jako u amplitudové obálky.

Filter Vypínatelná/zapínatelná sekce filtru. Filtruje signál po průchodu amplitudovou obálkou.

F. CutOff Zkratka pro Filter CutOff. Jedná se nastavení mezní frekvence filtru relativně k výšce tónu.

F. Vibrato Depth Reguluje hloubku modulace sinusovým nízkofrekvenčním oscilátorem.

F. Vibrato Rate Reguluje frekvenci nízkofrekvenčního oscilátoru, který moduluje mezní frekvenci tohoto filtru.

LFO Je zkratka pro Low Frequency Oscillator. Jedná se tedy o sekci nízkofrekvenčních oscilátorů, která je vypínatelná/zapínatelná. Lze vybrat průběh vlny a cíl modulace. A lze také ovlivňovat frekvenci LFO a hloubku modulace.

Freq. Je frekvence LFO. Maximální hodnota odpovídá hodnotě 10 Hz.

Depth Je hloubka modulace, která určuje, jak moc ovlivňuje cíl modulace. Pokud LFO moduluje frekvenci, tak maximální hodnota odpovídá dvěma půltónům. Pokud moduluje amplitudu, tak maximální hodnota ovlivňuje amplitudu v celém rozsahu, tedy například pila způsobí růst amplitudy z nuly do maxima.

Modulation Destination Jedná se o výběr cíle modulace, tedy jaký parametr bude LFO ovlivňovat. Lze vybrat z možností *Amplitude* a *Frequency*, tedy lze modulovat amplitudu nebo frekvenci signálu jdoucího z generátoru FM syntézy.

Wave Type Je výběr typu vlny LFO. Lze vybírat z možností *Sine*, *Triangle*, *Sawtooth* a *Square*. Tedy se jedná tvary vln sinusovka, trojúhelník, pila a obdélník.

Delay Je vypínatelná/zapínatelná sekce jednoduchého delay efektu.

Time Určuje čas zpoždění, tedy za jak dlouho se tón „ozve“.

Decay Určuje rychlost zániku ozvěny. Čím větší je decay, tím vícekrát se tón ozve. Při nulové hodnotě se ozve právě jednou.

Volume Regulace celkové hlasitosti syntezátoru. Současným použitím delay efektu a filtru se občas může stát, že setrvale znějící generovaná vlna může přesáhnout maximální hlasitost a výstup se zkreslí. Toto zkreslení lze odstranit právě zmenšením celkové hlasitosti.

Preset Pár přednastavených profilů k předvedení syntezátoru.

Base Octave Nastavuje základní oktávu klaviatury. Tedy snížení nebo zvýšení tohoto čísla snižuje nebo zvyšuje tóny klaviatury o oktávu.

C Výpočet koeficientů Chebyshevova filtru

Jedná se o obrázky skenované z knihy The Scientist and Engineer's Guide to Digital Signal Processing od autora jménem Steven Smith (tedy kniha [8]). Elektronická verze knihy je volně dostupná na `www.DSPguide.com`. Jazyk pseudokódu je BASIC.

```
100 'CHEBYSHEV FILTER-  RECURSION COEFFICIENT CALCULATION
110 '
120                                         'INITIALIZE VARIABLES
130 DIM A[22]                               'holds the "a" coefficients upon program completion
140 DIM B[22]                               'holds the "b" coefficients upon program completion
150 DIM TA[22]                              'internal use for combining stages
160 DIM TB[22]                              'internal use for combining stages
170 '
180 FOR I% = 0 TO 22
190   A[I%] = 0
200   B[I%] = 0
210 NEXT I%
220 '
230 A[2] = 1
240 B[2] = 1
250 PI = 3.14159265
260                                         'ENTER THE FOUR FILTER PARAMETERS
270 INPUT "Enter cutoff frequency  (0 to .5):  ", FC
280 INPUT "Enter  0 for LP,  1 for HP filter:  ", LH
290 INPUT "Enter percent ripple    (0 to 29):  ", PR
300 INPUT "Enter number of poles (2,4,...20):            ", NP
310 '
320 FOR P% = 1 TO NP/2                      'LOOP FOR EACH POLE-PAIR
330   '
340   GOSUB 1000                            'The subroutine in TABLE 20-5
350   '
360   FOR I% = 0 TO 22                      'Add coefficients to the cascade
370     TA[I%] = A[I%]
380     TB[I%] = B[I%]
390   NEXT I%
400   '
410   FOR I% = 2 TO 22
420     A[I%] = A0*TA[I%] + A1*TA[I%-1] + A2*TA[I%-2]
430     B[I%] =      TB[I%] - B1*TB[I%-1] - B2*TB[I%-2]
440   NEXT I%
450   '
460 NEXT P%
470 '
480 B[2] = 0                                'Finish combining coefficients
490 FOR I% = 0 TO 20
500   A[I%] = A[I%+2]
510   B[I%] = -B[I%+2]
520 NEXT I%
530 '
540 SA = 0                                  'NORMALIZE THE GAIN
550 SB = 0
560 FOR I% = 0 TO 20
570   IF LH = 0 THEN SA  =  SA    + A[I%]
580   IF LH = 0 THEN SB  =  SB    + B[I%]
590   IF LH = 1 THEN SA  =  SA    + A[I%] * (-1)^I%
600   IF LH = 1 THEN SB  =  SB    + B[I%] * (-1)^I%
610 NEXT I%
620 '
630 GAIN = SA / (1 - SB)
640 '
650 FOR I% = 0 TO 20
660   A[I%] = A[I%] / GAIN
670 NEXT I%
680 '                                       'The final recursion coefficients are in A[ ] and B[ ]
690 END
```

TABLE 20-4

Obrázek 13: Výpočet koeficientů Chebyshevova filtru, strana 1

```
1000 'THIS SUBROUTINE IS CALLED FROM TABLE 20-4,  LINE 340
1010 '
1020 '  Variables entering subroutine:        PI, FC, LH, PR, HP, P%
1030 '  Variables exiting subroutine:         A0, A1, A2, B1, B2
1040 '  Variables used internally:            RP, IP, ES, VX, KX, T, W, M, D, K,
1050 '                                        X0, X1, X2, Y1, Y2
1060 '
1070 '                                        'Calculate the pole location on the unit circle
1080 RP = -COS(PI/(NP*2) + (P%-1) * PI/NP)
1090 IP =  SIN(PI/(NP*2) + (P%-1) * PI/NP)
1100 '
1110 '                                        'Warp from a circle to an ellipse
1120 IF PR = 0 THEN GOTO 1210
1130 ES = SQR( (100 / (100-PR))^2 -1 )
1140 VX = (1/NP) * LOG( (1/ES) + SQR( (1/ES^2) + 1) )
1150 KX = (1/NP) * LOG( (1/ES) + SQR( (1/ES^2) - 1) )
1160 KX = (EXP(KX) + EXP(-KX))/2
1170 RP = RP * ( (EXP(VX) - EXP(-VX) ) /2 ) / KX
1180 IP  = IP * ( (EXP(VX) + EXP(-VX) ) /2 ) / KX
1190 '
1200 '                                        's-domain to z-domain conversion
1210 T  = 2 * TAN(1/2)
1220 W  = 2*PI*FC
1230 M  = RP^2 + IP^2
1240 D = 4 - 4*RP*T + M*T^2
1250 X0 = T^2/D
1260 X1 = 2*T^2/D
1270 X2 = T^2/D
1280 Y1 = (8 - 2*M*T^2)/D
1290 Y2 = (-4 - 4*RP*T - M*T^2)/D
1300 '
1310 '                                        'LP TO LP, or LP TO HP transform
1320 IF LH = 1 THEN K = -COS(W/2 + 1/2) / COS(W/2 - 1/2)
1330 IF LH = 0 THEN K =  SIN(1/2 - W/2) / SIN(1/2 + W/2)
1340 D = 1 + Y1*K - Y2*K^2
1350 A0 = (X0 - X1*K + X2*K^2)/D
1360 A1 = (-2*X0*K + X1 + X1*K^2 - 2*X2*K)/D
1370 A2 = (X0*K^2 - X1*K + X2)/D
1380 B1 = (2*K + Y1 + Y1*K^2 - 2*Y2*K)/D
1390 B2 = (-(K^2) - Y1*K + Y2)/D
1400 IF LH = 1 THEN A1 = -A1
1410 IF LH = 1 THEN B1 = -B1
1420 '
1430 RETURN
```

TABLE 20-5

TABLE 20-4 and 20-5
Program to calculate the "a" and "b" coefficients for Chebyshev recursive filters. In lines 270-300, four parameters are entered into the program. The cutoff frequency, FC, is expressed as a fraction of the sampling frequency, and therefore must be in the range: 0 to 0.5. The variable, LH, is set to a value of *one* for a high-pass filter, and *zero* for a low-pass filter. The value entered for PR must be in the range of 0 to 29, corresponding to 0 to 29% ripple in the filter's frequency response. The number of poles in the filter, entered in the variable NP, must be an even integer between 2 and 20. At the completion of the program, the "a" and "b" coefficients are stored in the arrays A[ ] and B[ ] (a_0 = A[0], a_1 = A[1], etc.). TABLE 20-5 is a subroutine called from line 340 of the main program. Six variables are passed to this subroutine, and five variables are returned. Table 20-6 (next page) contains two sets of data to help debug this subroutine. The functions: COS and SIN, use radians, not degrees. The function: LOG is the natural (base *e*) logarithm. Declaring all floating point variables (including the value of π) to be double precision will allow more poles to be used. Tables 20-1 and 20-2 were generated with this program and can be used to test for proper operation. Chapter 33 describes the mathematical operation of this program.

Obrázek 14: Výpočet koeficientů Chebyshevova filtru, strana 2

D Obsah CD/DVD

V podstatě jsou tu jen dvě složky, a to složka se zdrojovými kódy *BcSynth* a složka zkompilovaného programu *BcSynthGui_release.* Složka *BcSynthGui_release* obsahuje zkompilovaný program s .dll knihovnama, které jsou nutné k jeho běhu. Samotný program se pouští pomocí *SynthBcGui.exe*.

Složka *SynthBc* obsahuje solution projektu ve Visual Studio 2010. V této solution se nachází dva projekty, *SynthBc* a *SynthBcGui*. *SynthBc* projekt obsahuje v podstatě všechny napsané kódy zabývající se tvorbou zvuku syntezátoru, s výjimkou Generator.h, stk.h, stk.ccp, RtAudio.cpp, které jsou z open source knihovny The Synthesis ToolKit (STK). Základní části této knihovny jsou ve složce *stk*, která je ve složce solution *SynthBc*. Projekt *SynthBc* není určen ke kompilaci release verze, ani neobsahuje implementaci GUI. V jeho hlavní metodě jsou jen menší pokusy na implementovaných částech syntezátoru, které jsem testoval ještě před implementací GUI. Pro kompilování release verze je určen projekt *SynthBcGui*.

SynthBcGui projekt obsahuje implementaci uživatelského rozhraní pomocí knihovny Qt 5.0.2, kterou je třeba mít ke zkompilování nainstalovanou. Soubory se zdrojovými kódy jsou zde jen naimportované ze SynthBc projektu, tedy kromě samotných zdrojových kódů implementace GUI. Je zde také stažený open source plugin pro Qt, který vytváří klaviaturu. Více o použitých nástrojích na straně 40 v sekci Použité nástroje.